Deep Deinterlacing

Michael Bernasconi
Disney Research|Studios

Abdelaziz Djelouah
Disney Research|Studios

Sally Hattori
The Walt Disney Studios

Christopher Schroers
Disney Research|Studios

Written for presentation at the
SMPTE 2020 Annual Technical Conference & ***Exhibition***

Abstract. *With the increased demands in streaming services today, existing catalog contents are finding new exposures to large audiences. However, catalog contents and other old footage are generally only available in interlaced format. Interlacing has traditionally been used to double the perceived frame rate without consuming additional bandwidth. This allowed enhanced motion perception and reduced visible flicker. While old CRT displays can display interlaced video directly, modern TV displays typically use progressive scanning. This has more recently increased the interest in high quality deinterlacing algorithms. Deinterlacing can be considered as an ill-posed inverse problem with the goal of reconstructing an original input signal. As such it is challenging to solve, which becomes most apparent in the cases of large motion. Interestingly however, we can easily describe the degradation incurred through the subsampling strategy employed in interlacing, and therefore it is an ideal candidate for a fully supervised deep learning approach. Despite recent successes of machine learning for other tasks such as upscaling and denoising, deinterlacing has been explored relatively little and the early solutions that employ learning do not manage to consistently outperform existing deinterlacing methods already established in the industry. In this paper, we aim to close this gap by proposing a novel approach to deep video deinterlacing. Our approach addresses previous shortcomings and leverages temporal information more coherently. In addition to describing our architecture and training process in detail, we also include an ablation study that shows the impact of our individual architectural choices. Last but not least we also conduct a detailed objective evaluation comparing our approach to existing industry solutions as well as earlier learning based methods and show that we can consistently achieve significantly improved quality scores.*

Keywords. Deinterlace, Deep Learning.

1 Introduction

The concept of breaking a video into interlaced lines was first formulated and patented by German Telefunken engineer Fritz Schröter in 1930 [10]. Interlacing effectively is a subsampling scheme that strikes a balance between resolution and frame rate, by alternatively sampling rows from two consecutive frames. It was introduced to address the fact that TVs were not yet powerful enough to display progressive videos at a sufficient refresh rate due to their high bandwidth.
This worked well for many decades when played back on old CRT monitors, which can natively display interlaced content and make it perceptually appear as a coherent image sequence, aided by the afterglow of the display's phosphor. However, modern TVs use progressive scanning and are not able to properly display interlaced content requiring an explicit deinterlacing step. This and the fact that old interlaced catalog content is finding new audiences through streaming services has increased the interest in high quality deinterlacing algorithms. Especially also since the interlacing scheme can come in more complex forms in catalog content: In addition to subsampling, a variation of the interlacing scheme has also been used to increase the video frame rate in a process referred to as telecine. Here four frames are turned into five in a 2:3 pulldown to convert from 23.976Hz to 29.97Hz. Catalogue content may thus be composed of sections that are purely interlaced, telecined, or complex spatially varying compositions of both which increases the complexity of proper deinterlacing.
Due to its nature, interlacing generally works well for videos with little motion but begins to break down when the motion becomes larger. In such cases comb like artifacts introduced by interlacing can become apparent (see Figure 1). Existing commercial tools use methods that range from simple spatial interpolation, via directional dependent filtering, to motion compensated interpolation. These methods often achieve good results but we observed that they can create artefacts in complex situations and as such there is still an opportunity to improve existing deinterlacing results.
Interestingly, the methods used in the industry do not yet leverage the capabilities of deep learning although deep learning has shown impressive results in image enhancement tasks

Figure 1: Artifacts introduced by interlacing (left) and our reconstructed image (right). Photo Credits: pexels.com.

that are closely related to deinterlacing such as upscaling [5, 11], inpainting [9, 13] or denoising [8, 15].
Some of the aforementioned problems are actually blind, making it harder to design appropriate solutions [3, 7], while deinterlacing is perfectly suited for fully supervised training as the degradation process of interlacing is known and well defined. Therefore deinterlacing seems to be an ideal candidate for a deep learning based solution.
However, even in academia, these opportunities have only been explored very little. Recently, Zhu *et al.* [16] investigated using deep learning for the deinterlacing problem however the proposed solution is limited to processing a single frame (two fields) at a time and thus does not optimally leverage temporal information.
We propose a deep learning based solution that builds on recent advances in super-resolution and leverages temporal information by considering pixels from multiple fields. Our solution does not require to know the particular interlacing scheme used (e.g. standard interlacing, telecine, or combinations thereof). The experimental evaluation investigates multiple alternatives for example regarding architectural choices or the size of the temporal window to consider. We achieve state of the art results in our quantitative evaluation and demonstrate preferable visual results. Even though we did not specifically tune our method or the implementation for speed, we obtain roughly 10fps for deinterlacing NTSC content on a standard desktop machine with a recent GPU.
In summary, the contribution of the paper is a neural network architecture for the deinterlacing problem that leverages information from neighboring fields, an ablation study regarding several design choices, as well as state-of-the-art results.
Our paper is organised as follows: First we cover the most important related works. Then we describe our solution and conduct an ablation study before comparing to other existing methods and finally concluding.

2 Related Work

While there is a rich selection of methods for deinterlacing, deep learning based approaches have been explored relatively little. We will first list traditional approaches for deinterlacing and then discuss methods that employ learning.

Traditional methods are often categorized into field combination deinterlacing (weaving, blending, inverse telecine), field extension deinterlacing (line doubling, interpolation), and motion compensation deinterlacing. The latter are regarded as the most advanced algorithms as they try to predict the direction and the amount of image motion between subsequent fields in order to better blend the two fields together. They may employ algorithms similar to block motion compensation used in video compression. One of the most popular and state-of-the art algorithms is Yadif [2]. A summary of non machine learning based deinterlacing algorithms can be found in [4].

(Deep) Learning-Based Deinterlacing has been explored very little so far. Nnedi3 [1] is one of the very few tools using learning in the form of shallow neural networks for intra frame deinterlacing. To the best of our knowledge, real time deep video deinterlacing [16]

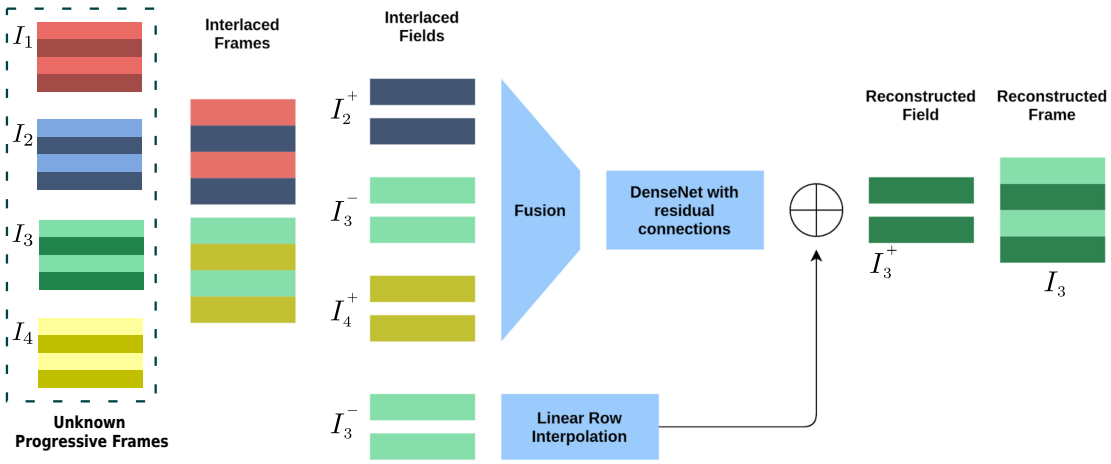

Figure 2: Overview of the proposed algorithm

is the first effort to address deinterlacing by means of *deep learning*. However, the method suffers from several drawbacks due to suboptimal use of temporal information, operating directly on interlaced inputs, and separately learning branches to fill in even and odd rows although this is essentially the same task.

3 Deep Deinterlacing

The objective of deinterlacing is to reconstruct an original unknown sequence of progressive frames from an interlaced input sequence. A progressive frame I_k consists of its odd and even rows, denoted as I_k^- and I_k^+, respectively. We will refer to these rows as fields. If we consider an interlaced sequence $\{I_1^-, I_2^+, I_3^-, I_4^+\}$, the objective is to recover the missing rows in order to reconstruct the original sequence $\{I_1, I_2, I_3, I_4\}$.

To deinterlace a sequence of frames, the first step in our model as shown in Figure 2 is separating the two fields corresponding to two different instants in time for each input frame. The two fields are ordered in time corresponding to the original video frames. Applying this process to an interlaced sequence yields a sequence of alternating top and bottom fields. Our goal is to find the missing half of the field sequence. Note that in order to perform this step it must be known whether the original video was interlaced in a top field first or a bottom field first way. The example in Figure 2 corresponds to a top field first setting and we will present our model in this setting before showing how bottom field first can be addressed by slightly transferring the same solution. Let's consider the case where the top field I_k^- is available. Our objective is to predict the bottom field I_k^+ for the same frame k, and we propose using a deep neural network to solve this problem.

The prediction function is denoted as f_θ where θ are the parameters of the neural network to be optimized:

$$I_k^+ = f_\theta(I_k^-, \mathcal{P}_{I_k^-}, \mathcal{N}_{I_k^-}). \tag{1}$$

Here $\mathcal{P}_{I_k^-}$ and $\mathcal{N}_{I_k^-}$ are the set of fields respectively preceding and following I_k. This allows for some flexibility in the considered temporal window. The size of the temporal window

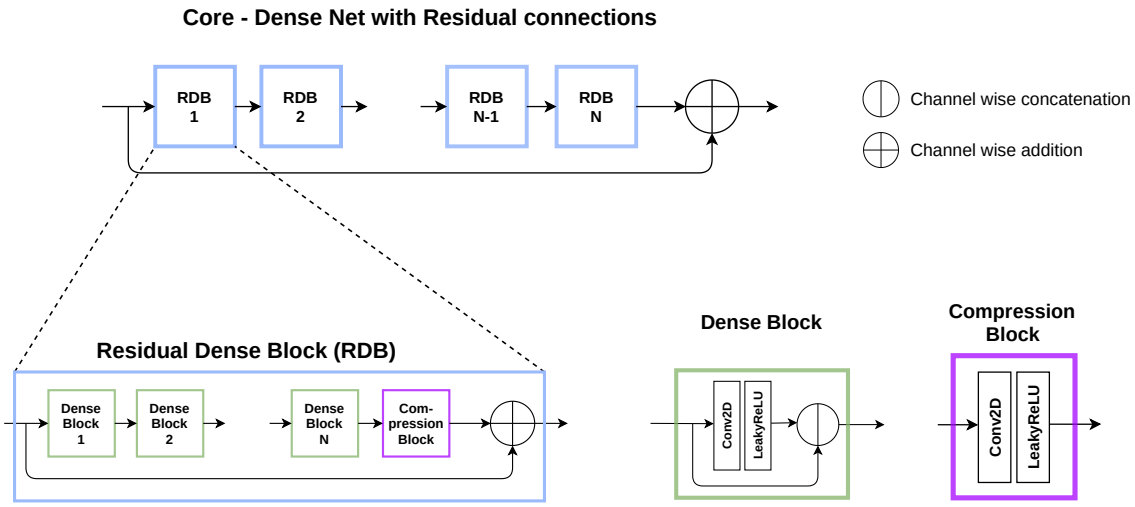

Figure 3: Detailed architecture for the deinterlacer

is denoted by N and it corresponds to the total number of fields used in the prediction. With $N = 1$, we are considering a prediction from I_k^- only. The case $N = 3$ is illustrated in Figure 2 where additional fields are taken into account. This corresponds to the case where $\mathcal{P}_{I_k^-}$ consists of the field I_{k-1}^+ and $\mathcal{N}_{I_k^-}$ consists of I_{k+1}^+. In our solution N only takes odd values and we have experimented with one, three, five and seven.

To predict the deinterlacing result, we first use linear interpolation of the known rows as an initial estimate of the missing rows. The neural network is then trained to predict the residual which is added to the initial estimate. The proposed architecture consists of two parts. The first one, *the fusion*, has the objective to produce an intermediate feature map from the different input fields. The second part, *the core deinterlacer*, computes the mapping to the RGB residual.

The solution we describe assumes that the prediction is made for the missing bottom field I_k^+, given the top field input I_k^-. To address deinterlacing in the other case, i.e. bottom fields, we first flip the fields vertically. This essentially turns them into a top field. The flipped field can then be deinterlaced and the output can be flipped vertically again. With this strategy, we can effectively transfer what was learned for top fields to the case of deinterlacing bottom fields to coherently treat both cases with a single trained network. Next we present the core of our deinterlacing architecture, the different fusion alternatives we considered and finally the training procedure.

3.1 Core Deinterlacing Architecture

The combination of residual networks (ResNets) and dense networks (DenseNets) has produced impressive results for related problems like super-resolution. Due to this, we use an architecture design inspired by our earlier work [11]. Figure 3 shows the deinterlacing architecture in detail. The most basic building blocks we use are the dense and compression blocks. These building elements are used to form the residual dense block which itself is

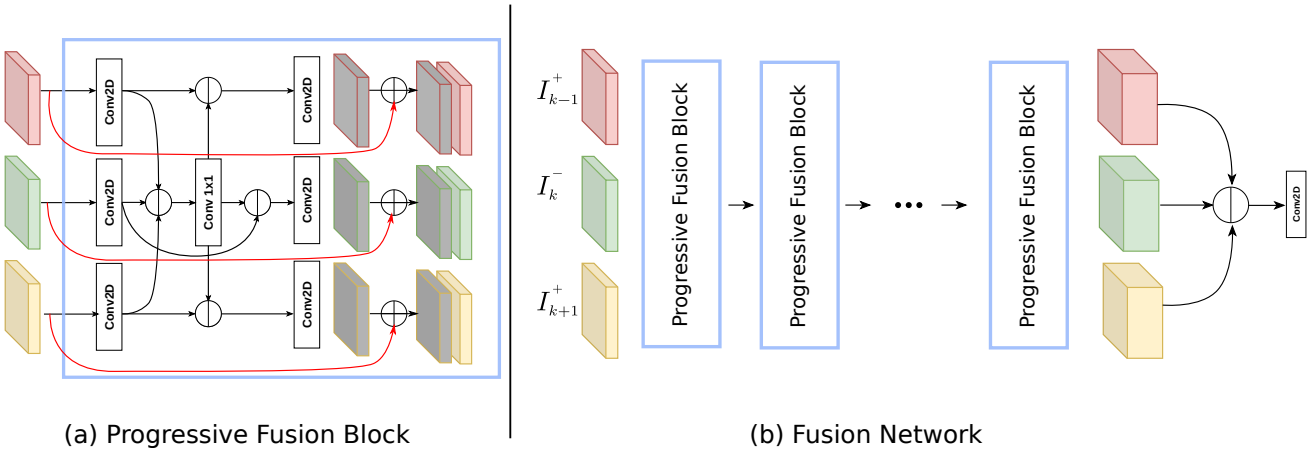

Figure 4: Details of the progressive fusion [14].

used to form our Dense Net with residual connection. To control the size of the network we can tune both the number of residual dense blocks (RDB) and the number of dense blocks in each RDB. In our experiments we explored different settings for the number of residual dense blocks M.

3.2 Multi-Field Fusion

Our core network takes a single feature map as input. Here we describe two different alternatives to obtain such a tensor from multiple input fields.

Direct Fusion is the simplest option where the fields are concatenated along the color channel dimension and directly provided to the deinterlacing neural network.

Progressive Fusion is a more complex solution to progressively extract and merge information between multiple fields. The details of the architecture are illustrated in Figure 3. It is similar in spirit to the Dense Net with residual connections used in the deinterlacing part. The difference here is that we maintain parallel branches, one for each input field, until the end where features are merged into a single tensor.

3.3 Training

The network is trained to minimize the objective function $\mathcal{L}$ over the dataset $\mathcal{D}$ consisting of interlaced frames

$$\theta^\star = \arg\min_\theta \mathbb{E}_{I_k \in \mathcal{D}} \left[\mathcal{L}\Big(f_\theta(I_k^-, \mathcal{P}_{I_k^-}, \mathcal{N}_{I_k^-}), \quad I_k^+ \Big) \right] \tag{2}$$

For the image loss we use the $\ell 2$-norm of pixel differences:

$$\mathcal{L}\Big(f_\theta(I_k^-, \mathcal{P}_{I_k^-}, \mathcal{N}_{I_k^-}), \quad I_k^+ \Big) = ||f_\theta(I_k^-, \mathcal{P}_{I_k^-}, \mathcal{N}_{I_k^-}) - \ I_k^+||_2 \tag{3}$$

4 Experimental Results

We evaluate our model on a custom dataset consisting of a wide variety of different video clips. The peak signal to noise ratio (PSNR) is used to measure the distortion between ground truth and reconstruction. Finally, we compare our method against Yadif [2] and Real-time Deep Video Deinterlacing (RDVD) [16]. All our models were trained on the Vimeo90K [12] dataset with a batch size of two and a learning rate of 0.0005 using the Adam optimizer [6]. The Vimeo90K dataset consists of roughly 90'000 sequences of seven frames with resolution 448x256. These are progressive sequences (our ground truth) from which we create the interlaced input used for training. We use the training / testing split that is provided with the dataset.

4.1 Ablation Study

In this section we aim to answer the following two questions. How many input fields are needed? And does progressive fusion yield any benefit over the simpler direct fusion?

Optimal number of input fields. To determine the optimal number of input fields our model is trained using direct fusion and one, three, five and seven input fields. For each case three and six RDB blocks are tested.

	1 Frame	3 Frames	5 Frames	7 Frames
3 Blocks	33.64	36.90	36.79	36.23
6 Blocks	33.50	**37.20**	37.07	36.64

Table 1: PSNR achieved by our method using varying numbers of input fields and RDB blocks on our custom data set. All models use direct fusion.

Table 1 shows that using multiple input fields yields much better results compared to using just a single input field. Further, using six instead of three RDB blocks only yields marginal gains compared to the performance difference between the single and multi field methods. Increasing the number of input fields beyond three does not yield any additional benefit.

Direct Fusion vs. Progressive Fusion. While direct fusion is simple it might not be able to fully extract the temporal information available in all input fields. Progressive fusion is a more sophisticated fusion technique which has shown promising results for related problems [14]. To see if progressive fusion yields any benefit over direct fusion we train our method using both techniques. To keep the number of parameters similar the direct fusion methods are trained using six RDB blocks while the progressive fusion method are trained using three RDB blocks.
Table 2 shows that progressive fusion does not yield better results than direct fusion. Additionally, progressive fusion comes at the cost of much higher memory consumption and slower runtime during both training and inference. We conclude that direct fusion is the better choice and will be used to compare our method against other state of the art methods.

	3 Frames	5 Frames
Direct Fusion	**37.20**	37.07
Progressive Fusion	36.79	36.23

Table 2: PSNR achieved by our method using direct and progressive fusion with similar numbers of parameters.

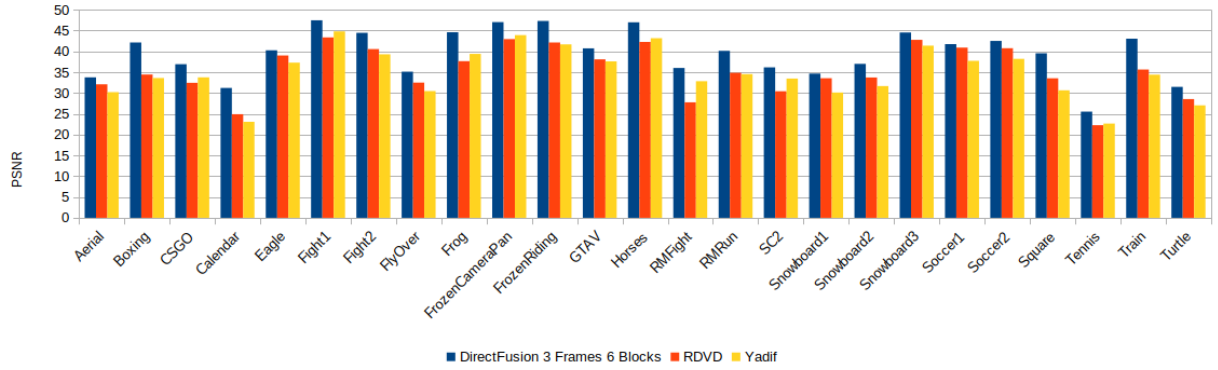

Figure 5: PSNR values achieved by different methods over our custom dataset.

4.2 Comparison to other methods

We compare our method against Yadif [2] and RDVD [16]. We have previously concluded that direct fusion is sufficient and no more than three input fields are required. Due to this we choose our direct fusion model with three input fields and six RDB blocks for all comparisons against other methods. First, we perform a numerical comparison between the methods by looking at the PSNR values achieved over our custom dataset.

Figure 5 shows our method outperforming both other methods in every single case. Often the gap in performance between our method and the others is quite large. To see if these large PSNR gains translate to better visual quality we take a closer look at two individual frames from our dataset.

Figure 6 shows that the improved PSNR values from our method translate to better visual quality. In the tennis and the calendar examples we see Yadif reconstruction the text and numbers poorly. The tennis example especially shows Yadif reaching it's limits when it comes to quick camera movement. Our method manages to reconstruct high quality images in both cases. The beach and turtle examples clearly show our method's strong performance when reconstructing thin details. Yadif and RDVD both introduce gaps into the palm tree's leafs and the white stripes on the turtle's head. Our method manages to produce continuous thin details in both cases.

In addition to yielding higher quality frames our method has another advantage compared to the other two. When looking at video sequences instead of individual frames our method produces drastically fewer flickering artifacts. The absence of flickering is essential for producing high quality video reconstructions. Unfortunately flickering cannot be measured or illustrated by comparing individual frames.

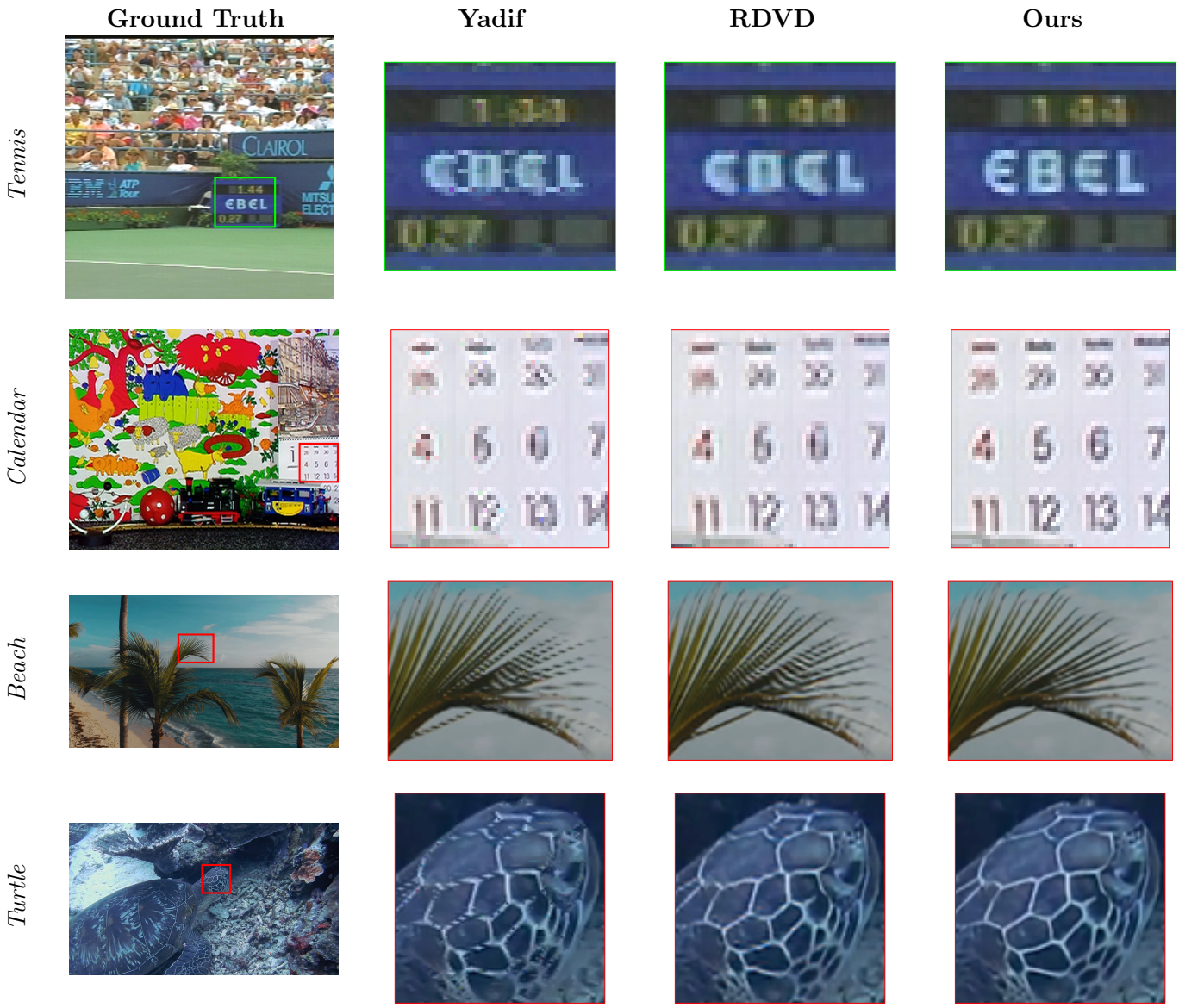

Figure 6: Visual comparison between Yadif, RDVD and our method. Our method clearly produces the best results in terms of visual quality. Photo Credits: *Beach* and *Turtle* from pexels.com

5 Conclusion

In this paper we introduced a new method for deinterlacing. Our proposed architecture is able to make use of more temporal information than previous deep learning based methods. By exploiting symmetries in the interlaced input our proposed architecture uses it's parameters efficiently. The proposed method surpasses current state-of-the-art techniques by a large margin in terms of PSNR as well as visual quality. The main advantages of our method are it's robustness and ability to use temporal information accurately. The accurate use of temporal information is especially noticeable in the reduction of flickering artifacts which other methods tend to produce. While we did investigate the progressive fusion technique it did not manage to produce higher quality results than simple direct fusion. Exploring more sophisticated fusion techniques, perhaps with attention layers, could be an interesting avenue for future research. A more powerful fusion technique could help the model to make use of even more temporal information which could yield even higher quality reconstructions.

References

[1] AviSynth. Nnedi3. Accessed: 2020-04-28.

[2] A. Balakhnin. Yadif. https://avisynth.org.ru/yadif/yadif.html, 2009.

[3] V. Cornillere, A. Djelouah, W. Yifan, O. Sorkine-Hornung, and C. Schroers. Blind image super-resolution with spatially variant degradations. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019.

[4] G. De Haan and E. B. Bellers. Deinterlacing-an overview. *Proceedings of the IEEE*, 86(9):1839–1857, Sep. 1998.

[5] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[7] A. Krull, T.-O. Buchholz, and F. Jug. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2019.

[8] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3587–3596, 2017.

[9] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[10] F. Schröter. Interlacing. Accessed: 2020-04-28.

[11] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers. A fully progressive approach to single-image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[12] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision (IJCV)*, 127(8):1106–1125, 2019.

[13] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6721–6729, 2017.

[14] P. Yi, Z. Wang, K. Jiang, J. Jiang, and J. Ma. Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3106–3115, 2019.

[15] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[16] H. Zhu, X. Liu, X. Mao, and T.-T. Wong. Real-time deep video deinterlacing. *arXiv preprint arXiv:1708.00187*, 2017.