Across Scales & Across Dimensions: Temporal Super-Resolution using Deep Internal Learning

Liad Pollak Zuckerman[⋆1] Eyal Naor[⋆1] George Pisha[⋆3] Shai Bagon[2] Michal Irani[1]

[1]Dept. of Computer Science and Applied Math, The Weizmann Institute of Science
[2] Weizmann Artificial Intelligence Center (WAIC)
[3]Technion, Israel Institute of Technology

Project Website: www.wisdom.weizmann.ac.il/~vision/DeepTemporalSR

Abstract. When a very fast dynamic event is recorded with a low-framerate camera, the resulting video suffers from severe motion blur (due to exposure time) and motion aliasing (due to low sampling rate in time). True Temporal Super-Resolution (TSR) is more than just Temporal-Interpolation (increasing framerate). It can also recover new high temporal frequencies beyond the temporal Nyquist limit of the input video, thus *resolving both motion-blur and motion-aliasing* – effects that temporal frame interpolation (as sophisticated as it may be) cannot undo. In this paper we propose a "Deep Internal Learning" approach for true TSR. We train a video-specific CNN on examples extracted directly from the low-framerate input video. Our method exploits the strong recurrence of small space-time patches inside a single video sequence, both within and across different spatio-temporal scales of the video. We further observe (for the first time) that small space-time patches recur also *across-dimensions* of the video sequence – i.e., by swapping the spatial and temporal dimensions. In particular, the higher spatial resolution of video frames provides strong examples as to how to increase the temporal resolution of that video. Such internal video-specific examples give rise to strong self-supervision, requiring no data but the input video itself. This results in ***Zero-Shot Temporal-SR*** of complex videos, which removes both motion blur and motion aliasing, outperforming previous supervised methods trained on external video datasets.

1 Introduction

The problem of upsampling video framerate has recently attracted much attention [2,15,9,16,24,14]. These methods perform high-quality Temporal Interpolation on *sharp videos* (no motion blur or motion aliasing). However, temporal-interpolation methods cannot undo motion blur, nor motion aliasing. This is a fundamental difference between Temporal Interpolation and Temporal Super-Resolution.

What is Temporal Super-Resolution (TSR)?

The *temporal resolution* of a video camera is determined by the frame-rate and by the exposure-time of the camera. These limit the maximal speed of dynamic events that can

⋆joint first authors.

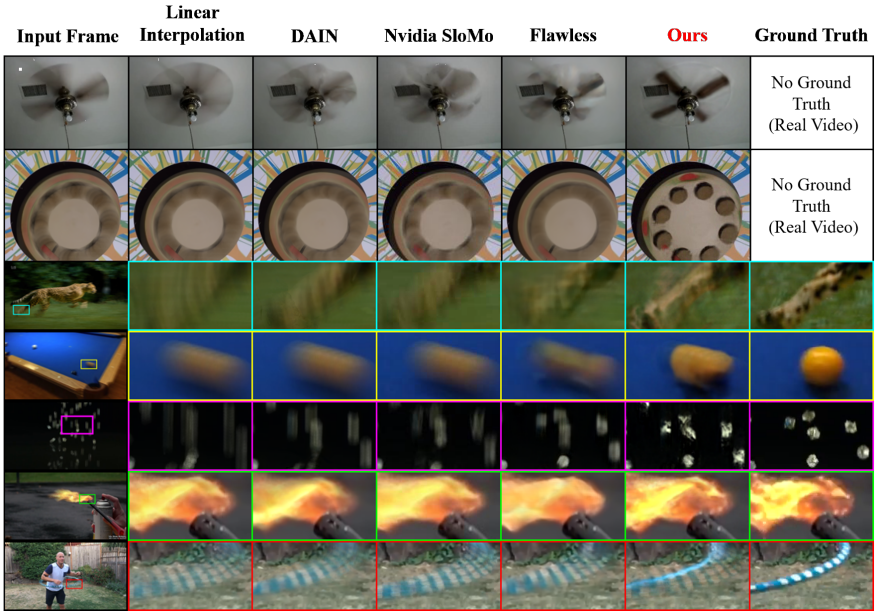

Fig. 1. Visual Comparison on TSR $\times$ 8. *We compared our method to state-of-the-art methods (DAIN[2], NVIDIA SloMo[9], Flawless[10]). Blurs of highly non-rigid objects (fire, water) pose a challenge to all methods. None of the competitors can resolve the motion blur or aliasing induced by the fast rotating fans. Our unsupervised TSR handles these better.* ***Please view videos in our project website to see the strong aliasing effects.***

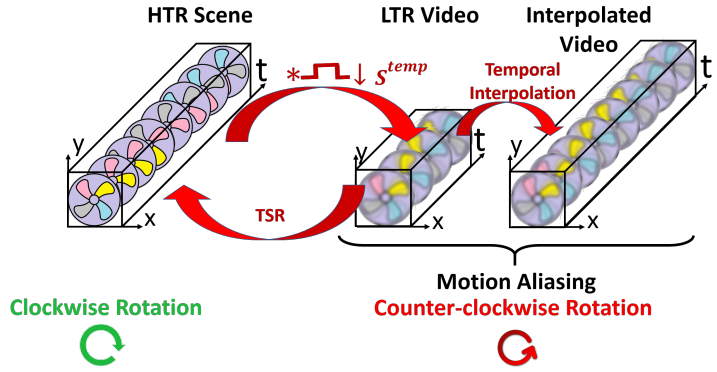

Fig. 2. Frame interpolation vs. Temporal-SR. *A fan is rotating clockwise fast, while recorded with a 'slow' camera. The resulting LTR video shows a blurry fan rotating in the wrong counter-clockwise direction. Temporal frame interpolation/upsampling methods cannot undo motion blur nor motion aliasing. They only add new blurry frames, while preserving the wrong aliased counter-clockwise motion.* ***Please see our project website to view these dynamic effects.*** *In contrast, true TSR not only increases the framerate, but also recovers the lost high temporal frequencies, thus resolving motion aliasing and blur (restoring the correct fan motion).*

be captured correctly in a video. *Temporal Super-Resolution* (TSR) aims to increase the framerate in order to unveil rapid dynamic events that occur *faster than the video-frame rate*, and are therefore invisible, or else seen incorrectly in the video sequence [19].

A low-temporal-resolution **(LTR)** video L, and its corresponding high-temporal-resolution **(HTR)** video H, are related by blur and subsampling in time:

$$L = (H * rect) \downarrow_{s^{temporal}}$$

where $rect$ is a rectangular temporal blur kernel induced by the exposure time.

For simplicity, we will assume here that the exposure time is equal to the time between consecutive frames. While this is a simplifying inaccurate assumption, it is still a useful one, as can be seen in our real video results (please view Fan video and the Rotating-Disk video in our project website). Note that the other extreme – the δ exposure model typically assumed by frame interpolation methods [2,9], is also inaccurate. The true exposure time is somewhere in between those two extremes.

When a very fast dynamic event is recorded with a "slow" camera, the resulting video suffers from severe motion blur and motion aliasing. Motion blur results from very large motions during exposure time (while the shutter is open), often resulting in distorted or unrecognizable shapes. Motion aliasing occurs when the recorded dynamic events have temporal frequencies beyond the Nyquist limit of the temporal sampling (framerate). Such an illustrative example is shown in Fig. 2. A fan rotating fast *clockwise*, is recorded with a "slow" camera. The resulting LTR video shows a *blurry fan* moving in the *wrong direction* – counter-clockwise.

Frame-interpolation methods [2,15,9,16,24,14] cannot undo motion blur nor motion aliasing. They only add new blurry frames, while preserving the wrong aliased counter-clockwise motion (illustrated in Fig. 2, and shown for real videos of a fan and a dotted wheel in Fig. 1 & full videos in the project website).

Methods for Video Deblurring (e.g., [7,23]) were proposed for removing motion blur from video sequences. These, however, do not increase the framerate, hence cannot resolve motion aliasing.

In contrast, true Temporal Super-Resolution (TSR) aims not only to increase the framerate and/or deblur the frames, but also to recover the lost high temporal frequencies beyond the Nyquist limit of the original framerate. "Defying" the Nyquist limit in the *temporal* domain is possible due to the motion blur in the *spatial* domain. Consider two cases: (i) A fan rotating clockwise fast, which due to temporal aliasing appears to rotate slowly counter-clockwise; and (ii) A fan rotating slowly counter-clockwise. When the exposure time is long (not δ), the fan in (i) has severe motion blur, while the fan in (ii) exhibits the same motion with no blur. Hence, while temporally (i) and (ii) are indistinguishable, spatially they are. Therefore, TSR can resolve both motion-aliasing and motion-blur, producing sharper frames, as well as the true motion of the fan/wheel (clockwise rotation, at correct speed). See results in Fig. 1, and full videos in the project website (motion aliasing is impossible to display in a still figure).

A new recent method, referred to in this paper as 'Flawless' [10], presented a Deep-Learning approach for performing *true TSR*. It trained on an external dataset containing video examples with motion blur and motion aliasing. Their method works very well on videos with similar characteristics to their training data – i.e., strong camera-induced

motion blur on mostly rigid scenes/objects. However, its performance deteriorates on natural videos of more complex dynamic scenes – highly non-rigid motions and severe motion-aliasing (see Fig. 1).

Failure to handle non-typical videos is not surprising — generating an inclusive video dataset containing all possible combinations of spatial appearances, scene dynamics, different motion speeds, different framerates, different blurs and different motion aliasing, is combinatorially infeasible.

In this work we propose to overcome these dataset-dependant limitations by replacing the External training with Internal training. Small space-time video patches have been shown to recur across different spatio-temporal scales of a single natural video sequence [18]. This strong internal video-prior was used by [18] for performing TSR from a single video (using Nearest-Neighbor patch search within the video). Here we exploit this property for training a Deep Fully Convolutional Neural Network (CNN) on examples extracted *directly from the LTR input video*. We build upon the paradigm of "Deep Internal Learning", first coined by [21]. They train a CNN solely on the input image, by exploiting the recurrence of small image-patches across scales in a single natural image [5]. This paradigm was successfully used for a variety of image-based applications [21,20,17]. Here we extend this paradigm, for the first time, to video data.

We further observe (for the first time) that *small space-time patches (ST-patches) recur also* ***across-dimensions*** *of the video sequence*, i.e., when swapping between the spatial and temporal dimensions (see Fig. 3). In particular, the higher *spatial resolution* of video frames provides strong examples as to how to increase the *temporal resolution* of that video (see Fig. 7.b). We exploit this recurrence of ST-patches across-dimensions (in addition to their traditional recurrence across video scales), to generate *video-specific training examples*, extracted directly from the input video. These are used to train a video-specific CNN, resulting in ***Zero-Shot Temporal-SR*** of complex videos, which resolves both motion blur and motion aliasing. It can handle videos with complex dynamics and highly non-rigid scenes (flickering fire, splashing water, etc.), that supervised methods trained on external video datasets cannot handle well.

Our contributions are several-fold:

- Extending "Deep Internal Learning" to video data.
- Observing the recurrence of data *across video dimensions* (by swapping space and time), and its implications to TSR.
- Zero-Shot TSR (no training examples are needed other than the input video).
- We show that internal training resolves motion blur and motion aliasing of complex dynamic scenes, better than externally-trained supervised methods.

2 Patch Recurrence across Dimensions

It was shown [18] that small Space-Time (ST) patches tend to repeat abundantly inside a video sequence, both within the input scale, as well as across coarser spatio-temporal video scales. Here we present a new observation ***ST-patches recur also across video dimensions***, i.e., when the spatial and temporal dimensions are swapped. Fig. 3 displays the space-time video volume (x-y-t) of a running cheetah. The video frames are the *spatial* x-y slices of this volume (marked in magenta). Each frame corresponds to the

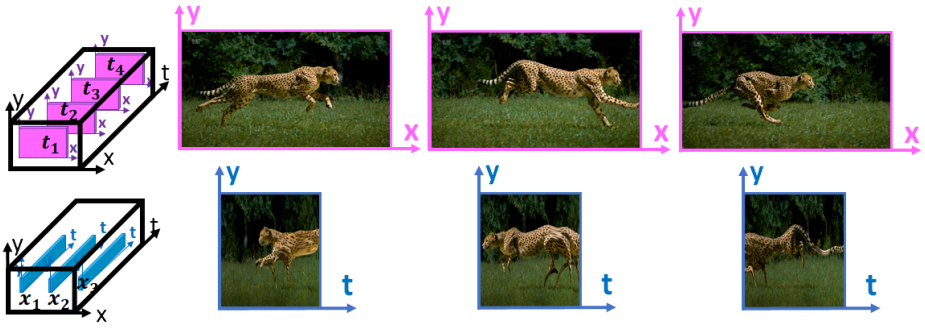

Fig. 3. Slices of the space-time video volume. *(Top:) xy slices = frames, (Bottom:) ty slices. The xy slices (video frames) provide high-resolution patch examples for the patches in the low-resolution ty slices. (See text for more details)*

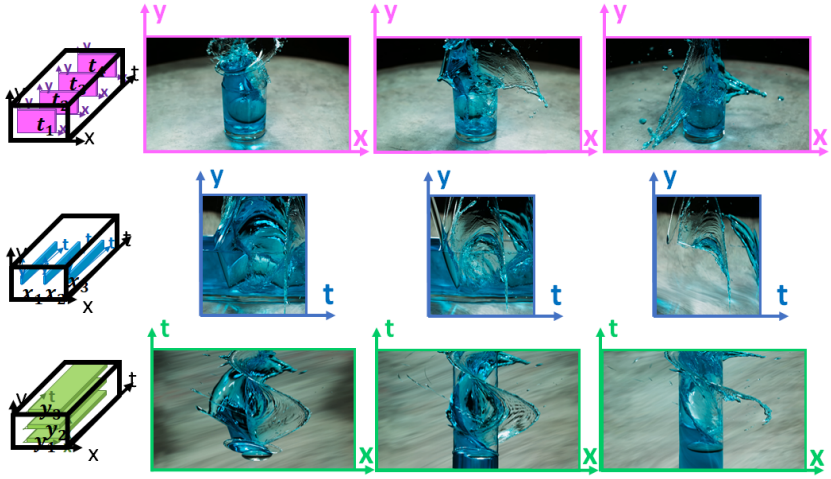

Fig. 4. Slices of the space-time video volume of a complex motion. *(Top to bottom:) xy slices, ty slices, xt slices. Note that patch similarities across-dimensions hold even for highly complex non-linear motions. See project website for videos of slices across different dimensions.*

plane (slice) of the video volume at time $t{=}t_i$. Swapping the spatial and the temporal dimensions, we can observe "frames" that capture the information in y-t slices ($x{=}x_i$ plane) or x-t slices ($y{=}y_i$ plane). Examples of such slices appear in Figs. 3 and 4 (green and blue slices). These slices can also be viewed dynamically, by flipping the video volume (turning the x-axis (or y-axis) to be the new t-axis), and then playing as a video. Such examples are found in the project website.

When an object moves fast, patches in x-t and y-t slices appear to be *low-resolution versions* of the higher-resolution x-y slices (traditional frames). Increasing the resolution of these x-t and y-t slices in t direction is the same as increasing the temporal resolution of the video. The *spatial* x-y video frames thus provide examples as to how to increase the *temporal* resolution of the x-t and y-t slices within the same video. In-

terestingly, when the object moves very slowly, patches in x-t and y-t slices appear as stretched versions of the patches in x-y frames, indicating that these temporal slices may provide examples as to how to increase the *spatial* resolution of the video frames. This however, is beyond the scope of the current paper.

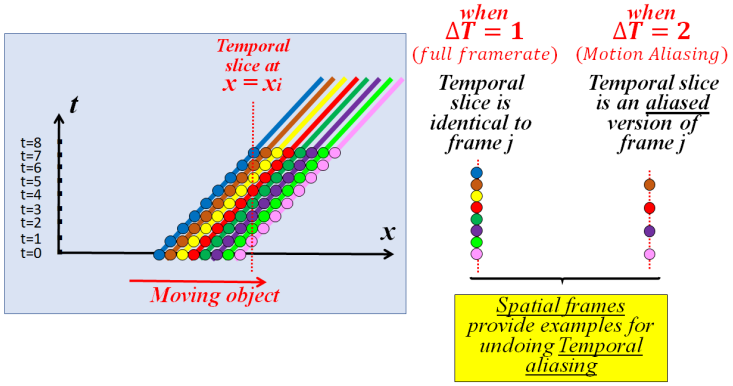

Fig. 5. 1D illustration of "across dimension" recurrence. *A 1D object moves to the right. When properly sampled in time (ΔT=1), temporal slices are similar to spatial slices (1D "frames"). However, when the temporal sampling rate is too low (ΔT=2), temporal slices are undersampled (aliased) versions of the spatial slices. Thus, spatial frames provide examples for undoing the temporal aliasing. (See text for more details.)*

Fig. 5 explains this phenomenon in a simplified "flat" world. A 1D object moves horizontally to the right with constant speed. The 2D space-time plane here (xt), is equivalent to the 3D space-time video volume (xyt) in the general case. If we look at a specific point $x{=}x_i$, the entire object passes through this location over time. Hence looking at the temporal slice through $x{=}x_i$ (here the slices are 1D lines), we can see the entire object emerging in that temporal slice. The resolution of the 1D temporal slice depends on the object's speed compared to the framerate. For example, if the object's speed is $1\ pixel/second$, then taking frames every $1second$ ($\Delta t\ =\ 1$) will show the entire object in the 1D temporal slice at $x{=}x_i$. However, if we sample slower in time (which is equivalent to a faster motion with the same framerate), the temporal slice at $x{=}x_i$ will now display an aliased version of the object (Fig. 5, on the right). In other words, the spatial frame at $t{=}t_0$ is a high-resolution version of the aliased temporal slice at $x{=}x_i$. The full-resolution *spatial* frame at $t{=}t_0$ thus teaches us how to *undo* the motion (temporal) aliasing of the *temporal* slice at $x{=}x_i$.

The same applies to the 3D video case. When a 2D object moves horizontally with constant speed, the y-t slice will contain a downscaled version of that object. The higher-resolution x-y frames teach how to undo that temporal aliasing in the y-t slice. Obviously, objects in natural videos do not necessarily move in a constant speed. This however is not a problem, since our network resolves only small space-time video patches, relying on the speed being constant only *locally* in space and time (e.g., within a $5{\times}5{\times}3$ space-time patch).

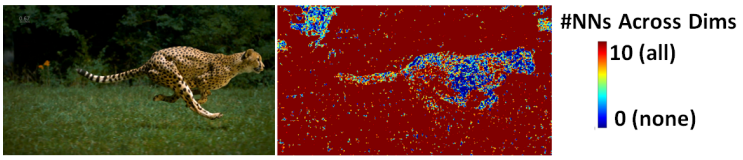

Fig. 6. Similarity Across-Dimensions: *Nearest-Neighbor (NN) heat map indicating the percent of best patch matches (out of 10), found "across-dimensions" vs. "within the same dimension". Red color indicates patches for which all 10 best matches were found "across dimension"; Blue indicates patches for which all 10 best matches were found "within the same dimension". As can be seen, a significant portion of the ST-patches found their best matches across dimensions.*

Shahar et. al [18] showed empirically that small 3D ST (Space-Time) patches tend to recur in a video sequence, within/across multiple spatio-temporal scales of the video. We refer to these recurrences as *'recurrence within the same dimension'* (i.e., no swap between the axes of the video volume). Patch *'recurrence across dimensions'* provides additional high-quality internal examples for temporal-SR. This is used *in addition* to the patch recurrence within the same dimension.

Fig. 6 visually conveys the strength of 'recurrence across dimensions' of small ST-patches in the Cheetah video, compared to their recurrence within the same dimension. Each $5\times5\times3$ patch in the original video searched for its top 10 approximate nearest-neighbors (using Patch-Match [4]). These best matches were searched in various scales, both within the same dimension (in the original video orientation, xyt), and across dimensions (tyx, by flipping the video volume so that the x-axis becomes the new t-axis). The colors indicate how many of these best matches were found across-dimension. Red color (100%) indicates patches for which all 10 best matches were found *across dimension*; Blue (0%) indicates patches for which all 10 best matches were found *within the same dimension*. The figure illustrates that a significant portion of the ST-patches found their best matches across dimensions (showing here one slice of the video volume). These tend to be patches with large motions – the background in this video (note that the background moves very fast, due to the fast camera motion which tracks the cheetah). Indeed, as explained above, patches with large motions can benefit the most from using the cross-dimension examples.

Both of these types of patch recurrences (within and across dimensions) are used to perform *Zero-Shot Temporal-SR* from a single video. Sec. 3 explains how to exploit these internal ST-patch recurrences to generate training examples from the input video alone. This allows to increase the framerate while undoing both motion blur and motion aliasing, by training a light and simple *video-specific CNN*.

Fig. 4 shows that patch recurrence across dimensions applies not only to simple linear motions, but also in videos with very complex motions. A ball falling into a liquid filled glass was recorded with a circuiting slow-motion (high framerate) camera. We can see x-t and y-t slices from that video contain similar patches as in the original x-y frames. Had this scene been recorded with a regular (low-framerate) camera, the video frames would have provided high-resolution examples for the lower temporal resolution of the x-t and y-t slices.

3 Generating an Internal Training Set

Low-temporal-resolution (LTR) & *High-temporal-resolution* (HTR) pairs of examples are extracted directly from the input video, giving rise to self-supervised training. These example pairs are used to train a relatively shallow fully convolutional network, which learns to increase the temporal resolution of the ST-patches *of this specific video*. Once trained, this video-specific CNN is applied to the input video, to generate a HTR output.

The rationale is: Small ST-patches in the input video recur in different space-time scales and different dimensions of the input video. Therefore, the same network that is trained to increase the temporal resolution of these ST-patches in other scales/dimensions of the video, will also be good for increasing their temporal-resolution in the *input* video itself. This is a similar logic to that of the ZSSR in images [21].

The creation of relevant training examples is thus crucial to the success of the learning process. In order for the CNN to generalize well to the input video, the LTR-HTR training examples should *bear resemblance* and have similar statistics of ST-patches as in the input video and its (unknown) HTR version. This process is explained next.

3.1 Example Pairs from "Same Dimension"

The first type of training examples makes use of similarity of small ST-patches across spatio-temporal scales of the video. As was observed in [18], and shown in Fig. 7.a:

- Downscaling the video frames spatially (e.g., using bicubic downscaling), causes edges to appear sharper and move slower (in pixels/frame). This generates ST-patches with *higher temporal resolution*.
- Blurring and sub-sampling a video in time (i.e., reducing the framerate and increasing the "exposure-time" by averaging frames), causes an increase in speed, blur, and motion aliasing. This generates ST-patches with *lower temporal resolution*. Since the "exposure-time" is a highly **non-ideal LPF** (its temporal support is $\leq$ than the gap between 2 frames), such temporal coarsening introduces additional motion aliasing.
- Donwscaling by the same scale-factor both in space and in time (the diagonal arrow in Fig. 7.a), preserves the same amount of speed and blur. This generates ST-patches with *same temporal resolution*.

Different combinations of spatio-temporal scales provide a variety of speeds, sizes, different degrees of motion blur and different degrees of motion aliasing. In particular, downscaling by the same scale-factor in space and in time (the diagonal arrow in Fig. 7.a), generates a variety of LTR videos, whose ST-patches are similar to those in the LTR input video, but for which their corresponding ground-truth HTR videos are known (the corresponding space-time volumes just above them in the space-time pyramid of Fig. 7.a).

Moreover, if the same object moves at different speeds in different parts of the video (such as in the rotating fan/wheel video in the the project website), the slow part of the motion provides examples how to undo the motion blur and motion aliasing in faster parts of the video. Such LTR-HTR example pairs are obtained from the bottom-left part of the space-time pyramid (below the diagonal in Fig. 7.a).

To further enrich the training-set with a variety of examples, we apply additional augmentations to the input video. These include mirror flips, rotations by $90°, 180°, 270°$,

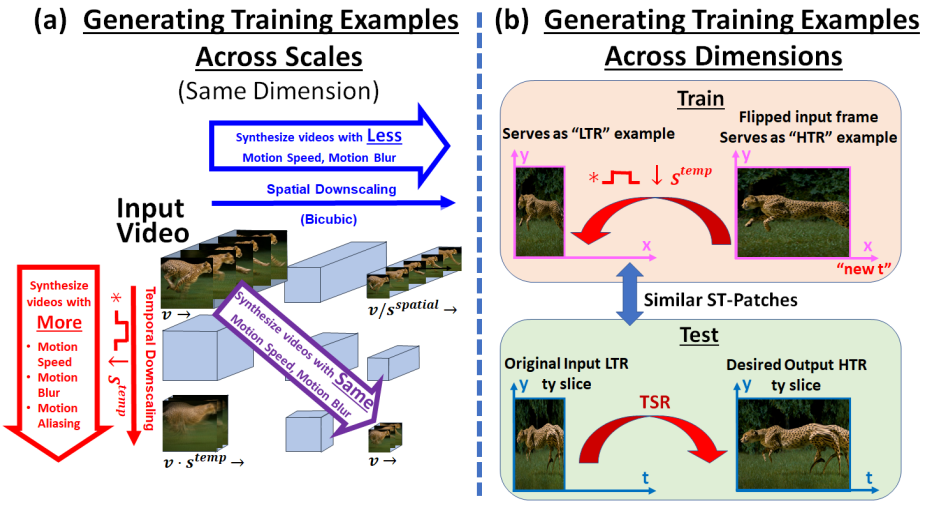

Fig. 7. Generating Internal Training Set. *(a) Different combinations of spatio-temporal scales provide a variety of speeds, sizes, different degrees of motion blur & aliasing. This generates a variety of LTR example videos, for which their corresponding ground-truth HTR videos are known (the space-time volumes just above them). (b) The xy video frames provide high-resolution examples for the ty and xt slices. Training examples can therefore be generated from these spatial frames, showing how to increase the temporal resolution.* ***Please see the video in our project website for a visual explanation and demonstration of those internal augmentations.***

as well as flipping the video in time. This is useful especially in the presence of chaotic non-rigid motions.

3.2 Example Pairs "Across Dimensions"

In order to make use of the similarity between small ST-patches across dimensions (see Sec. 2), we create additional training examples by rotating the 3D video volume – i.e., swapping the spatial and temporal dimensions of the video. Such swaps are applied to a variety of spatially (bicubically) downscaled versions of the input video. Once swapped, a variety of 1D temporal-downscalings (temporal rect) are applied to the *new* "temporal" dimension (originally the x-axis or y-axis). The pair of volumes before and after such "temporal" downscaling form our training pairs.

While at test time the network is applied to the input video in its original orientation (i.e., TSR is performed along the original t-axis), training the network on ST-patches with similarity across dimensions creates a richer training set and improves our results.

Here too, data augmentations are helpful (mirror flips, rotations, etc.). For example, if an object moves to the right (as in the Cheetah video), the y-t slices will bare resemblance to *mirror-reflected* versions of the original x-y frames (e.g., see the cheetah slices in Fig. 3).

In our current implementation, we use both types of training examples ('within-dimension' and 'across-dimensions') – typically with equal probability. Our experi-

ments have shown that in most videos, using both types of training examples is superior to using only one type while omitting the other type (see also ablation study in Sec. 5).

4 'Zero-Shot' Temporal-SR – The Algorithm

The repetition of small ST-patches inside the input video (aross scales and across dimensions), provide ample data for training. Such an internal training-set concisely captures the characteristic statistics of the given input video: its local spatial appearances, scene dynamics, motion speeds, etc. Moreover, such an internal training-set has relatively few "distracting" examples which are irrelevant to the specific task at hand. This is in stark contrast to the external training paradigm, where the vast majority of the training examples are irrelevant, and may even be harmful, for performing inference on a specific given video. This high quality training allows us to perform true TSR using a simple conv net without any bells and whistles; our model has no motion estimation nor optical flow components, nor does it use any complicated building blocks.

4.1 Architecture

A fully Convolutional Neural Network (CNN) efficiently calculates its output patch by patch. Each output pixel is a result of a calculation over a patch in the input video. The size of that patch is determined by the effective receptive field of the net [13]. Patch recurrence across scales and dimensions holds best for relatively small patches, hence we need to ascertain that the receptive field of our model is relatively small in size. Keeping our network and filters small (eight 3D conv layers, some with $3\times3\times3$ filters and some with $1\times3\times3$ filters, all with stride 1), we ensure working on small patches as required. Each of our 8 conv layers has 128 channels, followed by a ReLU activation. The input to the network is a temporally interpolated video (simple cubic interpolation), and the network learns only the residual between the interpolated LTR video to the target HTR video. Fig. 8.a provides a detailed description of our model.

At each iteration, a fixed sized space-time video crop of $36 \times 36 \times 16$ is randomly selected from the various internal augmentations (see Sec. 3). A crop is selected with probability proportional to its mean intensity gradient magnitude. This crop forms a HTR (High Temporal Resolution) space-time example. It is then blurred and subsampled by a factor of 2 *in time*, to generate an internal LTR-HTR training pair.

An ℓ_2 loss is computed on the recovered space-time outputs. We use an ADAM optimizer [12]. The learning rate is initially set to 10^{-4}, and is adaptively decreased according to the training procedure proposed in [21]. The training stops when the learning rate reaches 10^{-6}.

The advantage of such *video-specific* internal training is the adaptation of the network to the specific data at hand. The downside of such Internal-Learning is that it requires training the network from scratch for each new input video. Our network requires about 2 hours training time per video on a single Nvidia V100 GPU. Once trained, inference time at 720×1280 spatial resolution is roughly $1.7sec$ for each output frame.

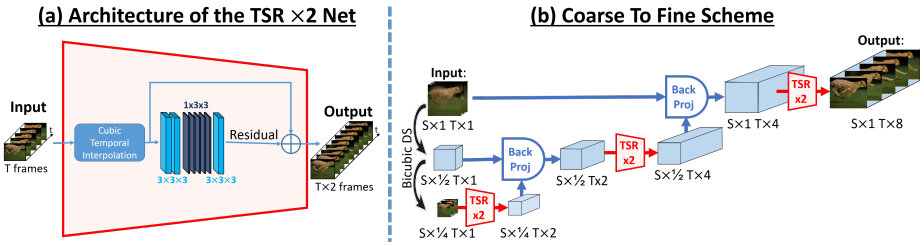

Fig. 8. Coarse to fine scheme and Architecture. ***(see text for details).***

4.2 Coarse-to-Fine Scheme (in Space & in Time)

Temporal-SR becomes complex when there are large motions and severe blur. As shown in Fig. 7.a, spatially downscaling the video results in smaller motions and less motion blur. Denoting the input video resolution by $S_{\times 1}T_{\times 1}$, our goal is to recover a video with $\times 8$ higher temporal resolution: $S_{\times 1}T_{\times 8}$. To perform our temporal-SR we use a coarse-to-fine approach (Fig. 8.b).

We start by training our network on a spatially downscaled version of the input video (typically $S_{\times 1/8}T_{\times 1}$, or $S_{\times 1/4}T_{\times 1}$ for spatially small videos). Fig. 8.b details a coarse-to-fine upscaling scheme from $S_{\times 1/4}T_{\times 1}$.The scheme to upscale from $S_{\times 1/8}T_{\times 1}$ includes an additional "Back-Projection" stage at the end. The network trains on this small video, learning to increase its temporal resolution by a factor of 2. Once trained, the network is applied to $S_{\times 1/8}T_{\times 1}$ to generate $S_{\times 1/8}T_{\times 2}$. We then use "Back-Projection"[1] [8] (both spatially and temporally), to increase the spatial resolution of the video by a factor of 2, resulting in $S_{\times 1/4}T_{\times 2}$. The spatial Back-Projection guarantees the *spatial* (bicubic) consistency of the resulting $S_{\times 1/4}T_{\times 2}$ with the spatially smaller $S_{\times 1/8}T_{\times 2}$, and its *temporal* (rect) consistency with the temporally coarser $S_{\times 1/4}T_{\times 1}$.

Now, since we increased both the spatial and temporal resolutions by the same factor ($\times 2$), the motion sizes and blurs in $S_{\times 1/4}T_{\times 2}$ remain similar in their characteristics to those in $S_{\times 1/8}T_{\times 1}$. This allows us to apply the same network again, as-is, to reach a higher temporal resolution: $S_{\times 1/4}T_{\times 4}$. We iterate through these two steps: increasing temporal resolution using our network, and subsequently increasing the spatial resolution via spatio-temporal Back-Projection, going up the diagonal in Fig. 7.a, until we reach the goal resolution of $S_{\times 1}T_{\times 8}$.

The recurring use of TSRx2 and "Back-Projection" accumulates errors. Fine-tuning at each scale is likely to improve our results, and also provide a richer set of training examples as we go up the coarse-to-fine scales. However, fine-tuning was not used in our current reported results due to the tradeoff in runtime.

5 Experiments & Results

True TSR (as opposed to simple frame interpolation) is mostly in-need when temporal information in the video is severely under-sampled and lost, resulting in motion alias-

[1] Don't confuse "Back-Projection" [8] with "backpropagation" [6].

ing. Similarly, very fast stochastic motions recorded within a long exposure time result in unrecognizable objects. To the best of our knowledge, a dataset of such low-quality (LTR) videos of complex dynamic scenes, along with their "ground truth" HTR videos, is not publicly available. Note that these are very different from datasets used by frame-interpolation methods (e.g., [24,11,22,1,3]), since these do not exhibit motion blur or motion aliasing, and hence are irrelevant for the task of TSR.

To address this lacuna, we curated a challenging dataset of 25 LTR videos of very complex fast dynamic scenes, "recorded" with a 'slow' (30 fps) video camera *with full inter-frame exposure time*. The dataset was generated from real complex videos recorded with high speed (mostly 240 fps) consumer cameras. The LTR videos were generated from our HTR 'ground-truth' videos by *blurring and sub-sampling them in time* (averaging every 8 frames). Since these 25 videos are quite long, they provide ample data (a very large number of frames) to compare and evaluate on. We further split our LTR dataset into two groups: (i) 13 extremely challenging videos, not only with severe motion blur, but also with severe motion aliasing and/or complex highly non-rigid motions (e.g., splashing water, flickerig fire, etc.); (ii) 12 less challenging videos, still with sever motion blur, but mostly rigid motions.

Fig. 1 displays a few such examples for TSR$\times$8. We compared our results (both visually and numerically) to the leading methods in the field (DAIN [2], NVIDIA SloMo [9] and Flawless[10]). As can be seen, complex dynamic scenes pose a challenge to all methods. Moreover, the rotating fan/wheel, which induce severe motion blur and severe motion aliasing, cannot be resolved by any of these methods. Not only are the recovered frames extremely distorted and blurry (which can be seen in the figure), they all recover a false direction of motion (counter-clockwise rotation), and with a wrong rotation speed. ***The reader is urged to view the videos in our project website in order to see these strong aliasing effects.*** Table 1 provides quantitative comparisons of all methods on our dataset – compared using PSNR, structural similarity (SSIM), and a perceptual measure (LPIPS[25]). The full detailed table of all 25 videos can be found in the project website. Since Flawless is restricted to $\times$10 temporal expansion (as opposed to the $\times$8 of all other methods), we ran it in a slightly different setting, so that their results could be compared to the same ground truth. Although most closely related to our work, we could not numerically compare to [18], due to its outdated dysfunctional software. Moreover, our end-to-end method is currently adapted to TSRx8, whereas their few published results are TSRx2 and TSRx4, hence we could not visually compare to them either (our TSRx2 network can currently train only on small (coarse) spatial video scales, whereas [18] applies SRx2 to their fine spatial scale.

The results in Table 1 indicate that sophisticated frame-interpolation methods (DAIN [2], NVIDIA SloMo [9]) are not adequate for the task of TSR, and are significantly inferior (-1 dB) on LTR videos compared to dedicated TSR methods (Ours and Flawless [10]). In fact, they are not much better (+0.5 dB) than plain intensity-based linear interpolation on those videos. Flawless and Ours provide comparable quantitative results on the dataset, even though Flawless is a pre-trained supervised method, whereas Ours is *unsupervised* and requires no prior training examples. Moreover, on the subset of extremely challenging videos (highly complex non-rigid motions), our Zero-Shot TSR outperforms the state-of-the-art externally trained Flawless [10]. We attribute this

		Ours	Flawless [10]	DAIN [2]	Nvidia SloMo [9]	linear interp.
Entire Dataset	PSNR [dB]	**28.27**	28.22	27.29	27.23	26.79
	SSIM	0.913	**0.918**	0.903	0.901	0.895
	LPIPS†[25]	0.194	**0.174**	0.214	0.214	0.231
Challenging Videos	PSNR [dB]	**28.05**	27.58	27.09	27.03	26.99
	SSIM	**0.922**	0.918	0.909	0.906	0.907
	LPIPS†[25]	**0.184**	0.188	0.208	0.205	0.212

† LPIPS (percetual *distance*) – lower is better. PSNR and SSIM – higher is better.

Table 1. Comparing TSR×8 results on our dataset. *When applied to challenging videos with severe motion blur and motion aliasing, sophisticated frame upsampling methods (Nvidia SlowMo and DAIN) score significantly lower. However, even methods trained to overcome such challenges (e.g., Flawless), but were pre-trained on an external dataset, struggle to compete on videos that deviate from the typical motions and dynamic behaviors they were trained on.*

to the fact that it is practically infeasible to generate an exhaustive enough external training set to cover the variety of all possible *non-rigid* motions. In contrast, highly relevant *video-specific* training examples are found internally, inside the LTR input video itself.

Since rigid motions are easier to model and capture in an external training set, Flawless provided high-quality results (better than ours) on the videos which are dominated by rigid motions. However, even in those videos, when focusing on the areas with non-rigid motions, our method visually outperforms the externally trained Flawless. While these non-rigid areas are smaller in those videos (hence have negligible effect on PSNR), they often tend to be the salient and more interesting regions in the frame. Such examples can be found in Fig. 1 (e.g., the billiard-ball and hoola-hoop examples), as well as in the videos in the project website.

5.1 Ablation Study

One of the important findings of this paper is the strong patch recurrence *across-dimensions*, and its implication on extracting useful internal training examples for TSR. To examine the power of such cross-dimension augmentations, we conducted an ablation study. Table 2 compares the performance of our network when: (i) Training only on examples from *same-dimension* ('Within'); (ii) Training only on examples *across-dimensions* ('Across'); (iii) Training each video on its best configuration – 'within', 'across', or on both.

Since our atomic TSRx2 network is trained only on a coarse spatial scale of the video, we performed the ablation study at that scale (hence the differences between the numeric values in Tables 2 and 1). This allowed us to isolate purely the effects of the choice of augmentations on the training, without the distracting effects of the subsequent spatial and temporal Back-Projection steps. Table 2 indicates that, on the average, the cross-dimension augmentations are more informative than the within (same-dimension) augmentations. However, since different videos have different preferences, training each video with its best within and/or across configuration provides an additional overall improvement in PSNR, SSIM and LPIPS (improvements are shown in blue parentheses in Table 2).

	Only Within	Only Across	Best of all configurations
PSNR [dB]	33.96	34.25 (✓0.28)	34.33 (✓0.37)
SSIM	0.962	0.964 (✓0.002)	0.965 (✓0.003)
LPIPS† [25]	0.035	0.033 (✓0.002)	0.032 (✓0.003)

† LPIPS (percetual *distance*) – lower is better. PSNR and SSIM – higher is better.

Table 2. Ablation study: 'Within' vs. 'Across' examples (mean values on dataset). *Average results of our atomic TSRx2 network, when trained on examples extracted from: (i) the same-dimension only ('Within'); (ii) across-dimensions only ('Across'); (iii) best configuration for each video – 'within', 'across', or both. The ablation results indicate that on the average, the cross-dimension augmentations are more informative than the within (same-dimension) augmentations, leading to an overall improvement in PSNR, SSIM and LPIPS (improvements are shown in blue parentheses). However, since different videos have different preferences, training each video with its best 'within' and/or 'across' configuration can provide an additional overall improvement in all 3 measures.*

This suggests that each video should ideally be paired with its best training configuration – a viable option with Internal training. For example, our video-specific ablation study indicated that videos with large uniform motions tend to benefit significantly more from cross-dimension training examples (e.g., the falling diamonds video in Fig. 1 and in the project website). In contrast, videos with gradually varying speeds or with rotating motions tend to benefit from within-dimension examples (e.g., the rotating fan video in Fig. 1 and in the project website). Such general video-specific preferences can be estimated per video by using very crude (even inaccurate) optical-flow estimation at very coarse spatial scales of the video. This is part of our future work. In the meantime, our default configuration randomly samples augmentations from both 'within' (same-dimension) and 'across-dimensions'.

6 Conclusion

We present an approach for Zero-Shot Temporal-SR, which requires no training examples other than the input test video. Training examples are extracted from coarser spatio-temporal scales of the input video, as well as from other video dimensions (by swapping space and time). Internal-Training adapts to the data-specific statistics of the input data. It is therefore more adapted to cope with new challenging (never-before-seen) data. Our approach can resolve motion blur and motion aliasing in very complex dynamic scenes, surpassing previous supervised methods trained on external video datasets.

Acknowledgments: The authors would like to thank Ben Feinstein for his invaluable help with getting the GPUs to run in a smooth and computationally efficient way. This project received funding from the European Research Council (ERC) Horizon 2020, grant No 788535. Additionally, supported by a research grant from the Carolito Stiftung. Dr Bagon is a Robin Chemers Neustein Artificial Intelligence Fellow.

References

1. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. International journal of computer vision **92**(1), 1–31 (2011) 12
2. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3703–3712 (2019) 1, 2, 3, 12, 13
3. Bao, W., Lai, W.S., Zhang, X., Gao, Z., Yang, M.H.: Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. IEEE transactions on pattern analysis and machine intelligence (2019) 12
4. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized patchmatch correspondence algorithm. In: European Conference on Computer Vision (ECCV). pp. 29–43. Springer (2010) 7
5. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: 2009 IEEE 12th international conference on computer vision (ICCV) (2009) 4
6. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016) 11
7. Hyun Kim, T., Mu Lee, K.: Generalized video deblurring for dynamic scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 3
8. Irani, M., Peleg, S.: Improving resolution by image registration. CVGIP: Graphical models and image processing (1991) 11
9. Jiang, H., Sun, D., Jampani, V., Yang, M.H., Learned-Miller, E., Kautz, J.: Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 1, 2, 3, 12, 13
10. Jin, M., Hu, Z., Favaro, P.: Learning to extract flawless slow motion from blurry videos. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 2, 3, 12, 13
11. Kiani Galoogahi, H., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: Proceedings of the IEEE International Conference on Computer Vision (CVPR) (2017) 12
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10
13. Luo, W., Li, Y., Urtasun, R., Zemel, R.: Understanding the effective receptive field in deep convolutional neural networks. In: Advances in neural information processing systems (NeurIPS) (2016) 10
14. Meyer, S., Djelouah, A., McWilliams, B., Sorkine-Hornung, A., Gross, M., Schroers, C.: Phasenet for video frame interpolation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 1, 3
15. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017) 1, 3
16. Peleg, T., Szekely, P., Sabo, D., Sendik, O.: IM-Net for high resolution video frame interpolation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 1, 3
17. Shaham, T.R., Dekel, T., Michaeli, T.: SinGAN: Learning a generative model from a single natural image. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 4

18. Shahar, O., Faktor, A., Irani, M.: Super-resolution from a single video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011) 4, 7, 8, 12
19. Shechtman, E., Caspi, Y., Irani, M.: Increasing space-time resolution in video. In: European Conference on Computer Vision (ECCV) (2002) 3
20. Shocher, A., Bagon, S., Isola, P., Irani, M.: InGAN: Capturing and remapping the "DNA" of a natural image. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 4
21. Shocher, A., Cohen, N., Irani, M.: "zero-shot" super-resolution using deep internal learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 4, 8, 10
22. Soomro, K., Zamir, A.R., Shah, M.: A dataset of 101 human action classes from videos in the wild. Center for Research in Computer Vision **2** (2012) 12
23. Su, S., Delbracio, M., Wang, J., Sapiro, G., Heidrich, W., Wang, O.: Deep video deblurring for hand-held cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1279–1288 (2017) 3
24. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. International Journal of Computer Vision (IJCV) **127**(8), 1106–1125 (2019) 1, 3, 12
25. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 12, 13, 14