Real-Time GPU-based Video Upscaling from Local Self Examples

Gilad Freedman Raanan Fattal
Hebrew University of Jerusalem

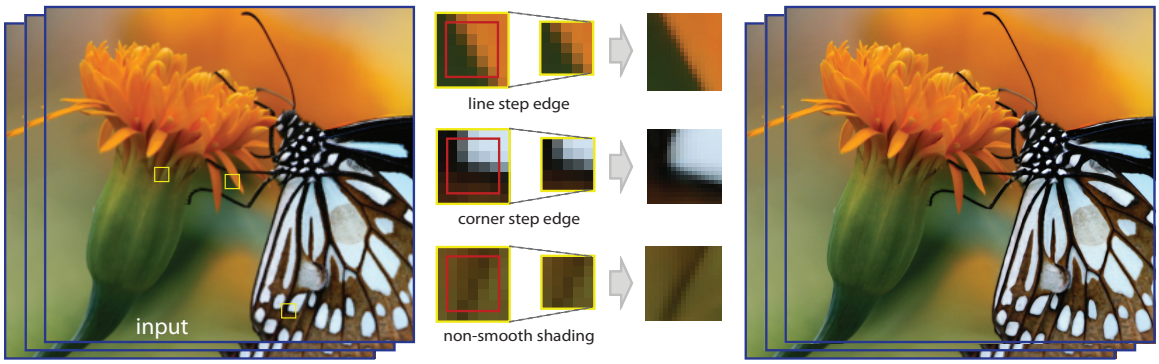

Figure 1: *Upscaling using local scale-similarity. The patches (yellow) when downscaled are very similar to their cropped version (red).*

Abstract. In our talk we describe a new high-quality and efficient GPU-based upscaling technique that extends existing example-based super-resolution frameworks in several respects. The new algorithm is inherently parallel and operates independently on small localized regions in the image. Leveraging these properties with a specialized GPU implementation allows to convert a video sequence in NTSC format (640×480 pixels) to full HD 1080p standard (1920×1080 pixels) in 24 fps, while meeting today's highest upscaling image quality.

Algorithm Overview. As depicted in Figure 2, given an input image I_1, we start off by interpolating it linearly to a finer grid. This *initial* upsampled image which we denote L_2 lacks a fraction of its upper frequency band, depending on the scaling factor. We predict this missing band using a non-parametric patch-based model that does not rely on external example databases but rather follows a *local self-similarity* assumption as follows. We start the prediction process by creating examples where we separate the input image into low and high frequency bands. This decomposition is done using a smoothing operator. We denote the smoothed input by L_1 and compute the high frequency band image by $H_1 = I_1 - L_1$. We then scan L_2, and for each of its patches p, we search in a small neighborhood $N(p)$ in L_1 centered at the same *normalized* coordinates. We pick the most similar patch, according to $q^* = argmin_q \|L_1(q) - L_2(p)\|_1$, and simply add $H_1(q^*)$ to $H(p)$ and average overlapping values. In our talk we provide more details on this scheme. A key component, responsible to the success of this framework is the construction of new dedicated linear interpolation, decimation and smoothing filtering, as we explain below.

Local Self-Similarity. As we mentioned above, the nearest patch search is *not* performed across the entire image, instead we exploit a *local* scale-invariance in images that allows to reproduce the native-resolution of various singularities in images, such as edges and corners, shown in Figure 1. In the talk we describe this regularity in natural images and show how highly relevant patches can be found at extremely localized regions in the downscaled image. In effect, this reduces the nearest-patch search efforts to just a small fraction, less than one-percent, compared to searching the entire image or using popular approximate nearest neighbor packages. We are certain this assumption can be used in the future for various other applications in computer graphics.

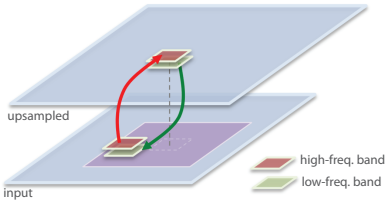

Figure 2: *Upscaling scheme. A patch of lower frequency band from the upsampled image is matched (green arrow) to its nearest patch within a local neighborhood in the low-passed input image (purple window). The upper frequency band of the matched patch in the input is taken (red arrow) to fill in the missing upper band in the output upsampled image.*

Non-Dyadic Filter Bank The local self-similarity assumption holds better for low magnification factors. Therefore, we upscale the image in several small increments, and to that end we developed a new dedicated filter bank. The construction of this filter bank follows principles commonly used in the construction of wavelets, yet *extends* the common dyadic wavelets to non-dyadic scaling.

Moreover, the new filters achieve an optimal exploitation of the input pixels and an increase in the visual realism, by maintaining consistency with the input image. While existing methods achieve this by solving large systems of linear equations, we design the filters such that they are nearly-biorthogonal. Thus, we obtain consistency though an explicit and therefore efficient computation, allowing our simple "copy-paste" high-frequency prediction scheme to achieve high quality results.

GPU Implementation. The operations performed in our algorithm, the linear interpolation, the smoothing, and the patch search are all local in terms of memory access. This property enables a very efficient GPU implementation, where every GPU core is assigned to a small part of the image, typically a window of 10-by-10 pixels. The cores keep the input and example pixels data in their fast local memory and thus avoid any access to the slower global device memory. We also execute our scheme on a subsampled set of pixels and exploit coherence in the example matches to further increase speed. These adaptations of the algorithms allow us to achieve a major speedup of two orders of magnitude, compared to a CPU implementations.

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007