Apparent Resolution Enhancement for Animations

Krzysztof Templin[1,2] Piotr Didyk[2] Tobias Ritschel[3] Elmar Eisemann[3]
Karol Myszkowski[2] Hans-Peter Seidel[2]

[1]University of Wrocław, Poland [2]MPI Informatik, Germany [3]Télécom ParisTech, France

Figure 1: *Highly detailed animations can be difficult to reproduce on current display devices. Here, four frames from animations used to evaluate our apparent resolution enhancement are shown: (a) rendering of a ball textured with text; (b) rendering of a detailed assembly of fibers; (c) a rendered short film ("Big Buck Bunny" © by Blender Foundation); (d) high-resolution content captured with a video camera.*

Abstract

Presenting the variety of high resolution images captured by high-quality devices, or generated on the computer, is challenging due to the limited resolution of current display devices. Our recent work addressed this problem by taking into account human perception. By applying a specific motion to a high-resolution image shown on a low-resolution display device, human eye tracking and integration could be exploited to achieve apparent resolution enhancement. To this end, the high-resolution image is decomposed into a sequence of temporally varying low-resolution images that are displayed at high refresh rates. However, this approach is limited to a specific class of simple or constant movements, i. e. "panning". In this work, we generalize this idea to arbitrary motions, as well as to videos with arbitrary motion flow. The resulting image sequences are compared to a range of other down-sampling methods.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image generation—display algorithms,viewing algorithms;

Keywords: image reconstruction, perception, eye tracking

1 Introduction

High-quality display and capture of images is important to give a faithful depiction of the natural world on a computer system. Unfortunately – although display devices as well as acquisition devices are constantly evolving – there is still a gap: not all images captured or generated can later be fully reproduced on the display devices available. This mismatch has recently gained attention and attempts were made to go beyond the physical capabilities of display devices using properties of the Human Visual System (HVS). One example is high-dynamic-range imaging, where current techniques are able to capture the real world luminance in a very accurate way. However, even the latest display devices are usually not capable of reproducing such content. In the context of 3D Vision, where the most popular 3D solutions are using shutter glasses, the low temporal resolution of the HVS is used to interleave images coming from the screen to the left and right eye.

In this work we are concerned with the fact, that many images and image sequences today exceed the spatial resolution of current displays. First, digital cameras produce still images and videos at a resolution an order of magnitude higher than displays can currently show. Additionally, super-sampling methods [Van Ouwerkerk 2006] can recover high-resolution spatial information from low-resolution videos by consolidating temporal variation. Second, efficient image synthesis algorithms and hardware [Akenine-Möller et al. 2008] made it possible to generate highly-detailed animations at even higher resolutions. In all those cases, down-sampling is required to present high-resolution content on low-resolution displays. Down-sampling however is prone to the loss of fine spatial details, such as hair, fur, other fiber structures or fine highlights.

While standard down-sampling methods treat each frame of an animation separately without taking temporal effects into account, it was shown by Didyk et al. [2010], how properties of the HVS can be beneficially exploited for moving images. The smooth pursuit eye movement (SPEM) mechanism can be used to induce subpixel shifts of the image projected on the retina. This, together with a high-refresh-rate display and careful derivation of displayed low-resolution frames, can lead to increased perceived resolution of the image. In their approach, SPEM was induced by artificially moving a still image with a constant velocity. In this work we will generalize this idea to arbitrary, spatially varying and temporally changing image motions, coming from the optical flow of an animation.

2 Previous Work

This section discusses related work that seeks to improve the viewing experience when observing computer displays. In particular, it is focused on work that involves human perception to improve reproduction of fine spatial details.

2.1 Spatial integration

One of the earliest ideas to increase apparent spatial resolution is to break the assumption that *R*, *G* and *B* channels form a single pixel [Benzschawel and Howard 1994]. So-called sub-pixel rendering exploits knowledge of sub-pixel layout, as found in modern liquid-crystal displays (LCDs) to allow for sub-pixel accurate detail localization. For LCD, the ClearType technology is one incarnation of this idea that is used to improve the depiction of text, which can be optimized for this purpose [Platt 2000]. However, only horizontal details are enhanced and best results are achieved for high-contrast black-and-white text. Application of sub-pixel rendering to complex images remains challenging, and care has to be taken to avoid introducing any new artifacts, such as moiré patterns [Messing and Kerofsky 2006; Klompenhouwer and de Haan 2003]. Hara and Shiramatsu [2000] observe that a special pixel-color mosaic extends the pass band of the image spectrum when moving an image with a specific velocity across the display. Some spatial sub-pixel layouts – such as an *RGGB*-mosaic – have shown to improve the perceived image quality. The standard $|RGB|RGB|\ldots$ arrangement, predominant in current LCD displays however, does not support such an improvement.

2.2 Temporal integration

Digital light processing (DLP) video projectors exploit the temporal integration in the HVS for color fusion. Different from spatial *RGB*-subpixel integration, they present the *RGB* color components sequentially in time with a temporal frequency over the perceivable flickering limit of the HVS. For some cases, such as rapid eye movements as found in saccades, spatial color disintegration ("rainbow effect") can become objectionable.

2.3 Image (re-)sampling

Displaying continuous or high-resolution input images on a finite, lower-resolution display is a sampling problem [Mitchell and Netravali 1988]. The display image is reconstructed by convolving the input image with a *reconstruction filter* for every output pixel. Popular reconstruction filters are Lanczos window and the family of cubic splines derived by Mitchell and Netravali [1988]. While these techniques were designed for static images, they are commonly applied to animated content as well. Our solution is different, as it takes into account multiple frames resulting in filtering that accords to the time-averaging nature of the HVS. By doing so, apparent spatial resolution will increase after temporal integration.

2.4 Optimal reconstruction

Recently a range of work has recast such sampling issues into optimization problems. Here, a number of spatially and temporally low-resolution images are optimized to combine into the perception of one high resolution image.

Wobulated projectors show multiple unique slightly-shifted sub-images using an opto-mechanical image shifter [Allen and Ulichney 2005], synchronized with the rapid sub-image projection to avoid flickering. This enhances the perceived image resolution and increases the perceived pixel area, which is otherwise limited by the door grid between physical pixels. Display supersampling achieves a similar effect by carefully-aligning multiple standard projectors [Damera-Venkata and Chang 2009]. They use an optimization for arbitrary (not raster-aligned) sub-pixel configurations. The present work is similar to [Allen and Ulichney 2005] and [Damera-Venkata and Chang 2009] in the sense that a high-resolution image is transformed into a set of low-resolution images, but we aim at a single desktop display or projector with a limited resolution and fixed pixel layout. We overcome these limitations by making use of smooth pursuit eye movement (SPEM) in arbitrary image sequences.

On the sensor-side, sub-pixel information acquired via subtle camera motion has proven useful in many applications, such as super-resolution reconstruction [Park et al. 2003] or video restoration [Tekalp 1995]. In these schemes, subpixel samples from subsequent frames are merged into explicitly reconstructed images, which, finally, are downsampled to match the display resolution. Note that our problem is, in some sense, an inverse problem. We do not need to reconstruct high frequency information because it is available in the original content. Instead, our task is to decompose high resolution images into low resolution subimages which are finally perceived as a high resolution image when displayed sequentially. Our approach avoids any explicit reconstruction and relies on perceptual processes to ensure detail enhancement.

Krapels et al. [2005] as well as Bijl et al. [2006] reported better object discrimination for subpixel camera panning than for corresponding static frames. Object discrimination improved regardless of the subpixel sensor motion rate, except for *critical velocities* [Tekalp 1995, C. 13] such as a one-pixel shift. A similar observation applies to rendering with supersampling where several images, rendered with slightly differing camera positions, are integrated in order to gain information.

Recently, it was shown by Didyk et al. [2010], how taking into account the HVS's SPEM, as well as its temporally integrating nature, the apparent resolution of a display can be enhanced for moving images. They proposed a simplified model of the temporal integration on the human retina, which allows them to predict the perceived image. In order to increase the apparent resolution of a given high-resolution image on a lower-resolution display device, they require to move an initially still image in a specific way. The motion can be general (arbitrary camera panning), but the method was demonstrated only for simpler linear motions. This allows to turn their model into an optimization problem, which takes the high resolution image as well as a characteristic of the motion as an input and, assuming SPEM, optimize for a sequence of low-resolution subimages to be shown on the output display. Displaying these images using a high-refresh-rate display creates a retinal image, whose resolution is higher than the resolution of the display device.

A similar idea was also proposed recently by Basu et al. [2009]. Their idea is also to introduce a certain motion to the displayed image and rely on the temporal domain by showing more information over time. However, in contrast to the previous method, they do not attempt to make the resulting image aliasing-free or to assure that temporal artifacts are not perceivable. They also propose a small circular path for the motion as the best choice, which at higher frame-rates can lead to simple averaging and no resolution gain. Whereas Didyk et al. [2010] *introduced* a motion for images in order to increase the apparent resolution, we show how to use *existing* motion in the scene (e. g., rendering, video animation) to achieve an enhancement as well. Therefore we will mostly rely on their findings extending them to arbitrary motion in a scene.

3 Model

Didyk et al. [2010] assume that after time T the response of a single receptor r moving over an image I is given by the following equation:

$$r = \int_0^T I(p(t),t)\,\mathrm{d}t, \tag{1}$$

where $p(t)$ denotes the position of the receptor at time t (cf. Fig. 2). While this assumption is not true in general [Van Hateren 2005], for sufficiently short T and periodic signal of frequency T^{-1} the equation holds due to the Talbot-Plateau law, which describes the HVS as a time-averaging sensor [Kalloniatis and Luu 2009].

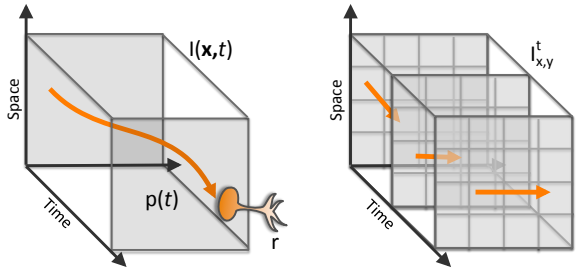

Figure 2: *Perceived integration of an animated image $I(x,t)$ (a space-time cube) by a receptor r moving along a continuous path $p(t)$. In the discrete case an animated image consists of discrete pixels at discrete frames of time with a discrete motion in every frame.*

Since I is a piecewise constant function of both time and position, the integral in Eq. 1 can be rewritten as a finite sum:

$$r = \sum_{i,j,k} w^k_{i,j} \cdot I^k_{i,j}, \tag{2}$$

where $I^k_{i,j}$ denotes the pixel at position (i,j) of the k-th subframe. The weights $w^k_{i,j}$ sum up to 1, and are proportional to the time spent by the receptor looking at the respective pixel, which can be formally stated as:

$$w^k_{i,j} = \frac{1}{|p|}\int_0^T \chi_{i,j}(p(t))\chi_k(t)\,\mathrm{d}t. \tag{3}$$

The characteristic function $\chi_{i,j}$ equals 1 when its argument indicates position covered by pixel (i,j), and $\chi_k(t)$ equals 1 if at time t the k-th frame is being displayed (cf. Fig. 3). The weight is normalized by the total length of the path, here denoted as $|p|$.

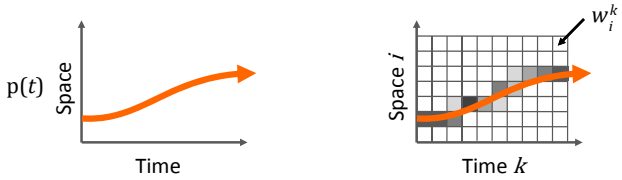

Figure 3: *Left: A receptor moving in (here 1D) space for different moments in time. Right: The weight w^k_i for every discrete pixel i in (here 1D) space and frame k in time, is proportional to the time the receptor is looking at this pixel.*

3.1 Solution

The formalism presented above is used by Didyk et al. [2010] to increase the apparent resolution of a 120 Hz display as follows: Suppose, that we want to show an image with a resolution 3 times higher than the resolution of the display. We assume that there is one-to-one correspondence between receptors and pixels of the high resolution image: one high-resolution pixel is seen by one receptor and one receptor sees one high-resolution pixel. In our solution, similarly to work by Didyk et al. [2010], we are not concerned with the size or the actual layout of photoreceptors in the human eye. Instead, we make the simplifying assumption that the number of receptors is equal to the number of pixels in the original high resolution image and that they are arranged on a grid. By moving the image on the display with a constant velocity we can initiate smooth pursuit movement of the observer's eyes. If the velocity, expressed in pixels of the high-resolution image per frame, is of the form (i,j), $i,j \in \mathbb{N}$, the position of the receptors during three frames of the animation will shift by an integer number of display pixels along both axes. Displaying the same three frames in a loop, shifted accordingly after every iteration, leads to a periodic signal reaching the receptors, with a frequency of 40 Hz. Although $T = 1/40\,\mathrm{s}$ is not short enough for the Talbot-Plateau law to hold under all circumstances [Kalloniatis and Luu 2009], it was shown to be sufficient in this scenario [Didyk et al. 2010].

The response of the receptors at any point of the animation can be computed according to Eq. 2. We seek to assign such values to the subframes' pixels, such that the response of the receptors is the same as for the high resolution image. Writing the display pixels from the three subframes as a column vector $\mathbf{x}$, and all the pixels in the high resolution image as a column vector $\mathbf{i}_\mathrm{h}$, we get the system $\mathsf{W}\mathbf{x} = \mathbf{i}_\mathrm{h}$, where the matrix W encodes the weights defined in Eq. 3, and each row corresponds to one receptor. As the dynamic range of the display should not be exceeded, every element x_i of $\mathbf{x}$ is constrained to $0 \le x_i \le 1$. Since this system is usually overdetermined, it does not have a solution. Instead, we find $\tilde{\mathbf{x}}$, minimizing the error with respect to the Euclidean norm: $\tilde{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} \|\mathsf{W}\mathbf{x} - \mathbf{i}_\mathrm{h}\|_2$.

3.2 Simple motion

By choosing a suitable motion, two important simplifications were made by Didyk et al. [2010]. First, the motion is assumed to be constant in time and space, i. e. it is a "camera pan": without any acceleration and with every part of the image moving at the same speed. Second, the motion is often set to be of the form (i,j), $i,j \in \mathbb{N}$, which we will call an *integer motion* in the high-resolution image space (i. e. integer multiples of 1/3 in the display space).

If these two conditions are fulfilled, all *triplets* (subsequences of 3 successive frames) in the animation are just spatially shifted copies of all previous or future triplets (cf. Fig. 4). We say, the problem has become *periodic*. In this case, the size of W and $\tilde{\mathbf{x}}$ is independent of the length of the animation, because it is sufficient to compute only a single triplet and repeat the sequence indefinitely, shifting it accordingly after every iteration.

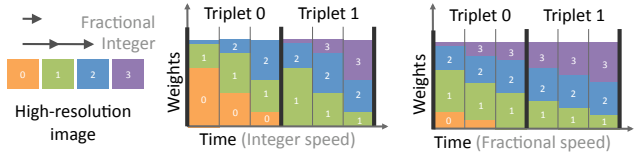

Figure 4: *Scrolling a high-resolution 4×1-image (left) over a single pixel. At integer velocities (middle), triplets of frames have the same weight (same areas in both triplets), only spatially shifted (color shift). At general velocities (right), no such periodicity is present (areas and hues different in both triplets).*

3.3 Non-integer motion

We will now generalize the approach of Didyk et al. [2010] to non-integer motions. The difficulty is, that after every triplet the position of the high-resolution image relative to the display-pixel grid changes, and the problem is therefore not periodic anymore (cf. Fig. 4, right). We extend the original system by adding equations for every possible triplet of the animation, i. e. for subframes $\{1,2,3\},\{2,3,4\},\{3,4,5\}$, etc. These additional equations are supposed to enforce animations, in which for every receptor and for every three consecutive subframes the integral computed according to Eq. 1 is as close as possible to the value of the corresponding high resolution pixel. In other words, our approach solves non-integer motions using many locally simple motions as done by Didyk et al. [2010]. It should be noted, however, that even if the conditions imposed by the additional equations are met for some receptors, the signal reaching them is not necessarily periodic.

One disadvantage of this approach is, that the size of W and $\tilde{\mathbf{x}}$ is now depending on the length of the animation, limiting, in practice, the effective length of the processable sequences.

3.4 General motion

As explained before, Didyk et al. [2010] achieve resolution enhancement by moving the whole image with a specific, spatially-invariant velocity enforcing a particular SPEM. We will now perform another generalization to spatially varying motion to achieve apparent resolution enhancement for arbitrary image sequences. As human observers are well-trained to track moving objects, we assume the tracking velocity to equal the spatially localized optical flow. Instead of enforcing a certain velocity for the entire image as done previously [Didyk et al. 2010], we consider the resolution enhancement problem locally in order to account for motion already present in a given animation.

Optical flow Horn and Schunk [1981] define optical flow as the distribution of apparent velocities of brightness patterns in an image. If one wants to focus on some detail of an image, they have to follow its apparent movement, thus it is reasonable to assume that between consecutive saccades SPEM closely follows the optical flow. We denote optical flow as a function $f\colon \mathbb{R}^2\times\mathbb{R}\to\mathbb{R}^2$, that applied to a location and a certain point in time returns the perceived velocity of the corresponding pattern. The function f can be obtained by a separate render pass in any advanced 3D animation authoring software (a so-called "motion vector pass") or – in the case of pre-existing animations or video material – using optical flow estimation methods [Horn and Schunck 1981; Zach et al. 2007].

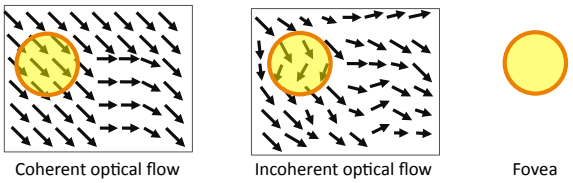

Figure 5: *Coherent (left) and incoherent (middle) optical flow. We assume that most of the time the optical flow does not vary over the foveal solid angle (right).*

We aim at animations where f is locally constant, both in space and time (cf. Fig. 5). Nevertheless f can vary substantially between distinct image regions and over time. Even though it is not possible to track the whole animation at once, we will optimize the subframes in each region according to its optical flow. For some parts of the animation this will violate the assumption that the eye movement matches the optical flow of the animation. However, when the eye does not follow the movement of a pattern, the perception of details in that region is already impaired, since the image is not stabilized on the retina. Thus, optimization in such cases should not lead to visible artifacts, and can improve the quality when the pattern is actually tracked by the observer. Additionally, as the HVS is not very sensitive to fine details outside the foveal region, which covers only ca. 2 visual degree, our assumptions hold perfectly if the optical flow is constant in the display area under the foveal solid angle.

Virtual receptors In the case of linear image motion, the positions being observed were well-defined, and formed a set of continuous paths. Two neighboring receptors that follow the optical flow can spread out, further, it is more difficult to track receptors because f can return arbitrary real-number shifts and the new position might in general not correspond to an accurate pixel position. Since we assume that f varies slowly, the velocity of the receptor could be approximated as a combination of its previous velocity and the velocities of the neighboring pixels, but this leads to a new problem: in every frame the number of possible locations to track increases. We solve those issues by introducing a new set of virtual receptors for every triplet of subframes. The purpose of the virtual receptors is to approximate the paths of the real receptors within a particular triplet. At the beginning of a triplet (i. e. in every frame) new virtual receptors appear at the positions of the high-resolution image pixels, then each virtual receptor starts moving with uniform velocity indicated by f, to disappear eventually after 3 frames. The response of the virtual receptor introduced in the k-th frame at the position $(x,y)^\mathsf{T}$ is defined as follows:

$$r^k_{x,y} = \int_0^3 I(p^k_{x,y}(t), k+t)\,\mathrm{d}t, \tag{4}$$

where the path $p^k_{x,y}$ is a segment given by:

$$p^k_{x,y}(t) = (x,y) + t\cdot f(x,y,k). \tag{5}$$

As before, we formulate a system of equations with every row corresponding to one virtual receptor, which we solve for minimal error with respect to the Euclidean norm.

It is worth noting that for the cases considered by Didyk et al. [2010] this formulation reduces to an equivalent optimization problem. Also the case of fractional velocities can be seen as a special case of this approach, with the exception that the grid of the virtual receptors at the moment of creation does not have to be pixel-aligned.

4 Implementation

In this section two different implementations of our model are presented: first, a CPU reference implementation (Sec. 4.1) and second, an efficient GPU implementation (Sec. 4.2).

4.1 CPU Implementation

The following iterative optimization scheme, similar to the gradient descent method is used in our C++ reference implementation:

$$\mathbf{x}^{(0)} = 0 \tag{6}$$

$$\mathbf{e}^{(i)} = \mathbf{i}_\mathrm{h} - \mathsf{W}\mathbf{x}^{(i)} \tag{7}$$

$$\mathbf{x}^{(i+1)} = \psi(\mathbf{x}^{(i)} + \mu\circ\mathsf{W}^\mathsf{T}\mathbf{e}^{(i)}) \tag{8}$$

$$\tilde{\mathbf{x}} = \mathbf{x}^n \tag{9}$$

where $\circ$ in Eq. 8 denotes component-wise multiplication, μ is a vector of scaling factors, and ψ is a clipping function, clamping every component of a vector to the range of $[0,1]$.

The algorithm proceeds by iteratively improving the current solution $\mathbf{x}^{(i)}$ as follows: Initially, all the solution pixels $\mathbf{x}^{(0)}$ are set to black (Eq. 6). Then, at each iteration, two steps are performed: error computation (Eq. 7) and error distribution (Eq. 8). Using the current solution $\mathbf{x}^{(i)}$ the receptor responses $\mathsf{W}\mathbf{x}^{(i)}$ are computed. The remaining errors $\mathbf{e}^{(i)}$ of this iteration are computed as the difference of the receptor responses $\mathsf{W}\mathbf{x}^{(i)}$ and the high-resolution animation pixels $\mathbf{i}_\mathrm{h}$. Next, the computed errors are distributed back onto the subframe pixels: for every subframe pixel, the weighted sum of the errors at the receptors it contributes to ($\mathsf{W}^\mathsf{T}\mathbf{e}^{(i)}$) is computed. Normalization is achieved, by setting μ_i^{-1} equal to the sum of the i-th row of W^T. Finally, the weighted average $\mu \circ \mathsf{W}^\mathsf{T}\mathbf{e}^{(i)}$ is added to the current solution $\mathbf{x}^{(i)}$, and clipped to the dynamic range of the display by the function ψ. The result $\tilde{\mathbf{x}}$ is the outcome of several iterations of this algorithm; in our experiments we used $n = 7$.

The matrix notation was introduced here for brevity, however W is sparse and its rows do not have to be constructed explicitly. In order to compute one entry in the error vector $\mathbf{e}^{(i)}$ we need to multiply the corresponding row of W by $\mathbf{x}^{(i)}$. However, all the sparse non-zero entries in one row of W can be enumerated efficiently, using only the respective receptor's starting position and its optical flow. Next, when the errors at the receptors are determined, for every receptor we iterate over all the contributing pixels again, to add the corrections. Since the normalization terms μ are not known before the loop over receptors ends, we first accumulate the weighted errors and the weights themselves in an auxiliary array, and then transfer them onto the solution pixels, clamping if necessary. Again, we avoid explicit construction of the rows of W^T.

4.2 GPU Implementation

Based on the CPU implementation from the previous section we will now introduce a parallel version, based on OpenGL shader programs (cf. Fig. 6). We assume the high resolution animation $\mathbf{i}_\mathrm{h}$ as well as its motion flow f (arrows in Fig. 6) to be given as floating-point input textures. No image data needs to be read back to the CPU at any point.

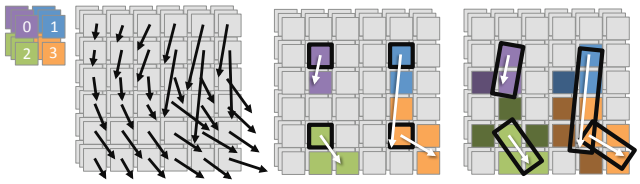

Figure 6: *One iteration of our GPU solver for a 6-by-6 pixel image shown on a 2-by-2 pixel display. From left to right, always showing only the first of three frames: (a): the current solution. (b): The motion flow. (c): Gathering of the error for four arbitrary example elements of the current solution. Every thread (solid black border) integrates over the motion flow (white arrows) in a for loop. (d): Error distribution for the same arbitrary elements. One line is drawn along the motion flow (white arrow) for each element (solid box), covering all non-zero weights, plus some additional pixels (darker colors).*

As described before, the initial solution $\mathbf{x}^{(0)}$ is set to black, but now stored in a renderable texture. Again, one iteration of the algorithm consist of two steps: error computation and error distribution.

The prediction $\mathsf{W}\mathbf{x}^{(i)}$ of how the current solution $\mathbf{x}^{(i)}$ is seen by the retina is computed, and then subtracted from the high-resolution input animation $\mathbf{i}_\mathrm{h}$ to produce $\mathbf{e}^{(i)}$. This is done in parallel for all receptors using a fragment shader.

The errors $\mathbf{e}^{(i)}$ are used in the second step of the iteration to improve the current solution. To this end, the error of each receptor is distributed across all solution pixels that contribute to it, proportional to the amount of time the receptor covers that pixel i. e. the weights defined in Eq. 3. In the CPU implementation, we took advantage of the fact, that we could randomly access memory to splat the error of each receptor to the pixels it sees. Now, for an efficient GPU implementation, line drawing is used to perform this step in parallel for all receptors. Each receptor along with its error and optical flow defines a line in the corresponding low resolution subimages. By drawing all the lines with additive blending we can compute the overall correction to every pixel in the current iteration. To efficiently draw the lines, a geometry shader is used. For every receptor the geometry shader produces three lines, one for each subframe, that start at the receptor's location and follow the optical flow f. When drawing the lines, a fragment shader computes the weights, i. e. how much time the receptor of this line spent in the corresponding pixel. This is done by computing the length of the line segment resulting from clipping the receptor's motion flow (a line equation) against the solution pixel's rectangle. While rasterizing, the RGB channels store the distributed error as a color and the alpha channel accumulates the error weights. In the final step, parallel over all solution pixels, RGB is normalized by the value stored in the alpha channel and clamped to the range of $[0,1]$.

4.3 Discussion

Performance As expected, the GPU implementation is several orders of magnitude faster. For an input sequence of 48 frames, with a resolution of 900×900 pixels, our C++ implementation needed 32 minutes to compute the optimized sequence of low resolution frames. Using a GPU, the same sequence can be computed in 13 seconds, which is 148 times faster compared to our CPU implementation.

Convergence Our algorithm closely follows the one used by Damera-Venkata and Chang [2009], with the exception that we use a scaling vector μ instead of a single scalar, so strictly speaking it is not the gradient descent method. While the procedure always converged for our problems, we have no formal proof of convergence. Intuitively, it can be considered as a localized version of the gradient descent method. In the special case of constant optical flow our "vector" variant is equivalent to the "scalar" one. We compared the performance of both methods for selected animations, and the results were practically the same. Additionally our method converged slightly faster.

5 Experiments

This section describes experiments conducted to measure the performance of our method. In Sec. 5.1 fractional movements of a static image are considered, while Sec. 5.2 describes experiments with general animations. Afterwards, a user study related to those animations is presented in Sec. 5.3.

All high-refresh rate image sequences were presented using 120 Hz Samsung SyncMaster 2233RZ or 120 Hz Acer GD235HZ displays. The respective image sequences as well as a Microsoft Windows / OpenGL application to display them are available as supplemental material.

5.1 Fractional velocities

We conducted a preliminary experiment with a static image moving with different fractional velocities to analyze dependence of aliasing on the velocity.

Animations were generated for the speeds of the form (x, y), where $x, y \in [0, 3]$ and $x \leq y$. The range was covered uniformly using steps of 0.1 which gave 495 different animations. The test image used was the 600×600 image shown in Fig. 7 (left). The image was scaled down 3 times using our method. The aliasing found in the resulting images was subjectively rated using a 7-grade scale, where 1 meant very little or no aliasing and 7 meant very heavy aliasing. The results are shown in Fig. 7 (right).

Although all the fractional velocities led to aliasing artifacts (in the form of jaggy edges or visible flickering), for some cases they were almost imperceivable.

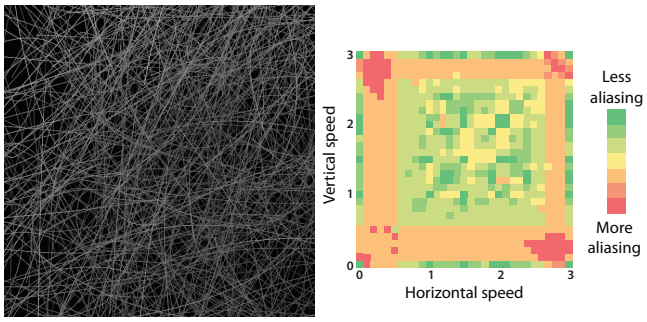

Figure 7: *Left: the image used for testing fractional velocities. Right: diagram showing the amount of aliasing as a function of velocity. The data was extrapolated by mirroring along the diagonal.*

5.2 General animations

To evaluate our method in the general case of arbitrary sequences we generated four animations of simple objects in Blender 2.49b and exported color and motion flow information in the floating point format. Two of these animations depicted rotating objects, with the axis of rotation parallel to the diagonal of the image: a ball with a text texture (Fig. 1, left) and a "hairy" ball (Fig. 1, middle-left). The remaining two animations showed periodic structures with the camera moving in a walk-through manner: two rows of fine-textured polyhedra (Fig. 8, left) and a text-textured tunnel (Fig. 8, right).

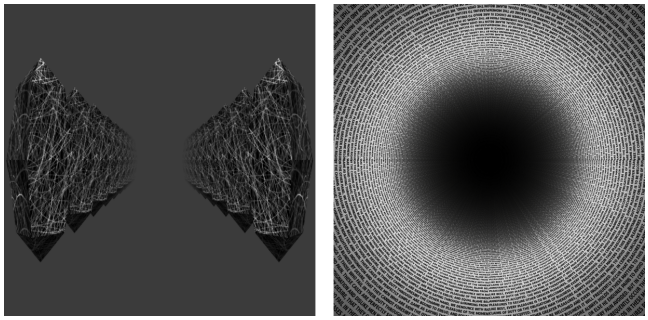

Figure 8: *The scenes used for testing. Two rows of fine-textured polyhedra (left) and a text-textured tunnel (right). In both cases the camera was moving in a walk-through manner.*

We also tested the scenario of unknown optical flow. To estimate the flow, we used an algorithm based on the TV-L_1 functional [Zach et al. 2007]. There were two test cases: the first one was an excerpt from the "Big Buck Bunny" film by the Blender Foundation (a 900×900 crop of frames 554–733). Since this film is intended for viewing at 24 fps, we chose the opening scene, which – thanks to the slow camera movement and static environment – can be easily observed even after speeding it up 5 times. The other test case was a video sequence of a printed text, sliding along the diagonal. The material was shot with a hand-held consumer camera.

We compared the results of our algorithm with simple frame-by-frame downsampling using Lanczos window as a reconstruction filter, with kernels of radius 3, 4, 5 and 6.

All the results share some common observations. First, Lanczos-6 was always more blurry, and Lanczos-3 resulted in stronger aliasing artifacts than our optimization method. Second, in all the results there were some nicely reproduced regions, and regions with visible aliasing. Finally, the detail level of our method was comparable to Lanczos-4, however the latter resulted in more pronounced aliasing.

We noticed, that the sharpness of the text-textured objects (the ball and the tunnel) was slightly better for our method than for Lanczos-4. The "hairy" ball test revealed one short-coming of our method – around the fibers little artifacts could be seen, probably originating from the discontinuities or inaccuracies of the optical flow. Interestingly, the level of artifacts related to the optical flow estimation errors was very low.

5.3 User Study

Additionaly, we conducted a user study to compare our method with Lanczos filtering. The solutions obtained using our method were again compared to the standard downsampling method: Lanczos filtering with a kernel size corresponding to performed decimation as well as downsampling using smaller kernels.

In our study we wanted to show, that in most cases our solution is preferred over Lanczos filtering with the kernel radius adjusted manually. Fourteen participants took part in the experiment. For each tested animation the study was performed in two steps. First, the original sequence was shown to the subject with four downsampled versions using different Lanczos kernels, and they were asked to choose the most suitable one. In general this is a demanding task and as shown in [Mitchell and Netravali 1988] the quality of downsampled content is subjective due to the trade-off between spatial detail reproduction and spatial aliasing. Therefore, we asked the subjects to choose the kernel size that in the best way reproduced the appearance of the original sequence in terms of detail visibility, taking also into account possible temporal aliasing problems.

In the second step, the subject was asked to compare the previously chosen animation to the sequence computed using our method. Both sequences where shown in random order with the reference aside. Similarly to the first part, the subject had to choose the method that reproduces the original animation better. Additionally, we ran the same study but this time downsampled versions of animations were upscaled in order to match their size with the reference sequence. This not only helps people match the appearance of two comparing animations, but also simulates bigger pixel sizes, which would usually occur for larger TV sets. The results of this study are shown in Tab. 1.

In this scenario, our solution was not always preferred over Lanczos filtering. To show the advantage coming from using our approach, we computed a global ranking from the obtained data. This was possible, as we already had information about how many times our solution was chosen and which size of the Lanczos filtering was preferred. The ranking indicates in how many cases each of the compared methods was preferred over all the others (Tab. 2). It

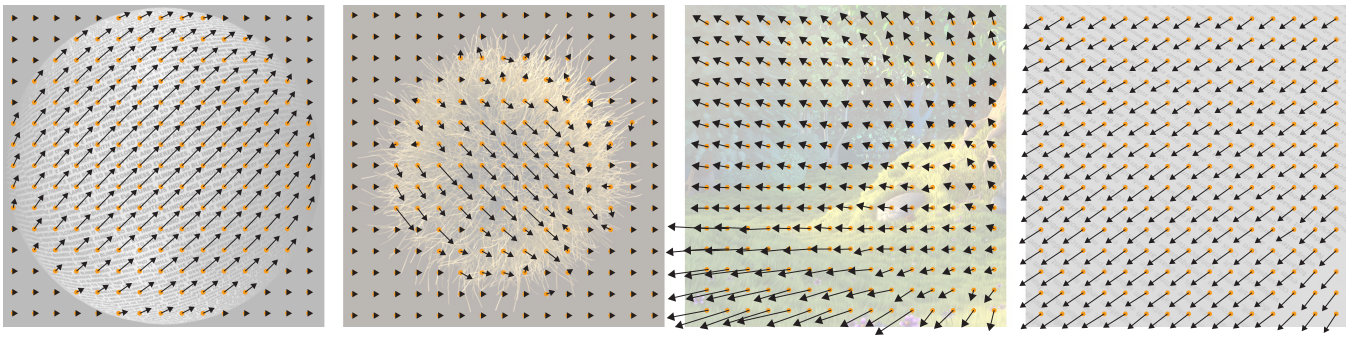

Figure 9: *The optical flow of the test animations.*

	Step 1	Step 2 (Our)
Lanczos 3	5 % (10 %)	75 % (56 %)
Lanczos 4	14 % (18 %)	58 % (53 %)
Lanczos 5	45 % (45 %)	54 % (58 %)
Lanczos 6	36 % (27 %)	30 % (27 %)

Table 1: *The results of the user study. The first column indicates the Lanczos kernel size. The second column (Step 1) contains the percentage of all the cases in which subjects chose the corresponding Lanczos filtering in the first step of the procedure. The last column shows in how many cases our solution was preferred comparing to the previously chosen Lanczos method. For example, in 14 % of all cases Lanczos-4 was chosen as the best size of the Lanczos filter, and in 58 % of them it was ranked lower than our solution, which constitutes 8 % of all the tests. The numbers in brackets correspond to the second version of the study, where the downsampled animations were later upscaled.*

turned out that our method was preferred in 47 % (60 % for the second version) of all cases.

Method	Preference
Lanczos 3	2 % (1 %)
Lanczos 4	6 % (3 %)
Lanczos 5	20 % (17 %)
Lanczos 6	25 % (19 %)
Our	47 % (60 %)

Table 2: *The global ranking computed from the data obtained in the user study. Percentage in brackets corresponds to the comparison with upscaled versions of downsampled animations.*

6 Discussion

For static image movement, Didyk et al. [2010] used filters of radius 6, while scaling down by a factor of 3. The reason was to prevent aliasing in the individual frames: standard decimation by factor of 3 using Lanczos filtering requires a kernel radius equal to 6 [Turkowski 1990]. For moving images however, we noticed that by using smaller kernel sizes we could often achieve very similar results to those obtained with the method introduced by Didyk et al. [2010]. Whereas filtering with the standard kernel size will generally remove the frequencies that cannot be reproduced in a static frame, smaller kernels – similarly to our approach – can leave some aliasing in the individual frames which may result in higher quality after temporal averaging. This led to the decision to include also smaller filters in further comparisons. Filtering with smaller kernels is similar to the idea proposed by Basu et al. [2009]. It also corresponds to the fact that the perfect filtering does not exist and it is usually a matter of taste which filter to use [Mitchell and Netravali 1988].

The first experiment showed, that our method cannot produce fully alias-free animations for static images moving with fractional velocities. However, the level of artifacts varies significantly from case to case, and for some velocities aliasing problems are almost imperceivable. Some examples of those velocities are: $(1.0, 1.5), (1.8, 1.8)$ or $(1.2, 2.4)$. Moreover, it is worth noting that fractional velocities can be problematic also for the Lanczos reconstruction method. If we are aiming at a flicker-free animation, the radius of the kernel often has to be extended beyond 6, and this leads to a heavily blurred image.

The study showed that our method is on average preferred over all the others considered in our study. We did not make any special effort in order to remove possible temporal fluctuations, therefore people, for whom the temporal coherence is the most important issue, will usually choose the kernel size 6 for Lanczos filtering in the first step, and subsequently rank it higher in the second step due to small temporal fluctuations of our method. This can explain why in the tests where Lanczos with kernel 6 was chosen, only in 30 % (27 %) cases it was later ranked lower than our solution. On the other hand, it turned out that many people appreciate a more detailed reconstruction at the price of a small temporal fluctuation, which in our method is comparable with the method that uses Lanczos filtering with the kernel size of 5. Because our solution offers more detailed reconstruction compared to Lanczos-5 as well as significantly less temporal artifacts than methods with smaller kernels, the participants showed preference of our solution over Lanczos filtering with the kernel size smaller than 6. As a future work it will be interesting to include a flickering reduction method to possibly match the temporal fluctuation of our method with those in standard Lanczos filtering with kernel size 6, keeping at the same time the detail reproduction high.

An interesting aspect of the proposed algorithm is its influence on readability of small-sized text. Didyk et al. [2010] proved that for scrolling text their method results in increased readability compared to Lanczos-6 downsampling. We observed a similar effect in the tests with text-textured objects. Our method produced significantly more legible results than Lanczos-6 and Lanczos-5. Superiority of our method in comparison with smaller kernels was not obvious, however it produced less artifacts.

It can be disputed, if using optical flow estimation methods is reasonable, because of their inherent inaccuracy. However, this decision can be supported to some extent by the following reasoning: if optical flow estimation gives poor results, the actual flow of the animation is probably complicated. But in such cases it is also hard to track the patterns and our perception of details in the animation is impaired.

7 Conclusion

This work introduced a technique to enhance the depiction of fine details in arbitrarily animated image sequences on high-refresh rate displays. Based on two substantial generalizations of a recent perceptual image enhancement approach [Didyk et al. 2010], optimization for content with arbitrary, spatially varying optical flow became possible. Benefits as well as limitations of the approach were analyzed in a perceptual study, that shows how an improvement is possible for a range of speeds and how to trade detail reproduction and aliasing in time and space. An efficient GPU implementation allows us to produce such enhanced image sequences at near-interactive computation times in practice.

There is a range of interesting avenues for further research. First, it will be interesting to see what enhancements are possible with even higher refresh rates, dynamic range, display sizes etc. For example higher refresh-rate displays may potentially solve in many cases the problem of temporal fluctuations, improving the quality of our method. Second, the optimization would ideally be performed in real-time for HD content, which is not yet in reach of our implementation and hardware. Such solution could be used along with an eye tracker to provide accurate information about the eye movement as well as allow performing local resolution enhancement. Third, we assume a regular and discrete layout of pixels and receptors to be given initially, whereas a raytracer can produce a non-regular sampling and could compute an arbitrary-resolution sampling when required. Along those lines, adaptive sampling in image synthesis could be steered by our approach to produce more samples if the optimization can use them and save computational time where the high-resolution image can not be reproduced anyway. The bigger picture of improving apparent quality on more general "displays" using on-line optimization for human perception, remains an exciting direction of further research: moving displays; moving observer; other reproduction techniques such as printing, stereo (anaglyph, shutter, polarization), holography and haptics as well as their mutual combination.

Acknowledgements

We would like to thank the anonymous reviewers for helpful comments and Kaustubh Patil (MPI-Informatik) for proofreading.

References

AKENINE-MÖLLER, T., HAINES, E., AND HOFFMAN, N. 2008. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA.

ALLEN, W., AND ULICHNEY, R. 2005. Wobulation: Doubling the addressed resolution of projection displays. In *Proc. Symp. Digest of Technical Papers (SID)*, vol. 47.4 of *The Society for Information Display*, 1514–1517.

BASU, S., AND BAUDISCH, P. 2009. *System and process for increasing the apparent resolution of a display (US Patent 7548662)*.

BENZSCHAWEL, J. L., AND HOWARD, W. E. 1994. *Method of and apparatus for displaying a multicolor image (US Patent 5341153)*.

BIJL, P., SCHUTTE, K., AND HOGERVORST, M. 2006. Applicability of TOD, MTDP, MRT and DMRT for dynamic image enhancement techniques. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 6207, SPIE.

DAMERA-VENKATA, N., AND CHANG, N. L. 2009. Display supersampling. *ACM Trans. Graph. 28*, 1, 9:1–9:19.

DIDYK, P., EISEMANN, E., RITSCHEL, T., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2010. Apparent display resolution enhancement for moving images. *ACM Trans. Graph. 29* (July), 113:1–113:8.

HARA, Z., AND SHIRAMATSU, N. 2000. Improvement in the picture quality of moving pictures for matrix displays. *J. SID 8*, 2, 129–137.

HORN, B. K. P., AND SCHUNCK, B. G. 1981. Determining optical flow. *Artifical Intelligence 17*, 185–203.

KALLONIATIS, M., AND LUU, C., 2009. Temporal resolution. http://webvision.med.utah.edu/temporal.html.

KLOMPENHOUWER, M. A., AND DE HAAN, G. 2003. Subpixel image scaling for color-matrix displays. *J. SID 11*, 1, 99–108.

KRAPELS, K., DRIGGERS, R. G., AND TEANEY, B. 2005. Target-acquisition performance in undersampled infrared imagers: static imagery to motion video. *Applied Optics 44*, 33, 7055–7061.

MESSING, D. S., AND KEROFSKY, L. J. 2006. Using optimal rendering to visually mask defective subpixels. In *Human Vision and Electr. Imaging XI*, vol. 6057 of *SPIE Proc. Series*, SPIE, 236–247.

MITCHELL, D. P., AND NETRAVALI, A. N. 1988. Reconstruction filters in computer-graphics. *Proc. SIGGRAPH 22*, 4, 221–228.

PARK, S., PARK, M., AND KANG, M. 2003. Super-resolution image reconstruction: A technical overview. *IEEE Signal Processing Magazine 20*, 3, 21–36.

PLATT, J. 2000. Optimal filtering for patterned displays. *Signal Processing Letters, IEEE 7*, 7, 179 –181.

TEKALP, A. 1995. *Digital Video Processing*. Prentice Hall.

TURKOWSKI, K., 1990. Turkowski filters for common resampling tasks.

VAN HATEREN, J. H. 2005. A cellular and molecular model of response kinetics and adaptation in primate cones and horizontal cells. *J. Vision 5*, 4, 331–347.

VAN OUWERKERK, J. 2006. Image super-resolution survey. *Image and Vision Computing 24*, 10, 1039–1052.

ZACH, C., POCK, T., AND BISCHOF, H. 2007. A duality based approach for realtime TV-L_1 optical flow. In *I. Ann. Symp. German Association Patt. Recogn.*, German Association Patt. Recogn., 214–223.