Fast Multi-Resolution Image Operations in the Wavelet Domain

Iddo Drori[*] Dani Lischinski[†]
School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel

October 4, 2001

Abstract

A wide class of operations on images can be performed directly in the wavelet domain by operating on coefficients of the wavelet transforms of the images and other matrices defined by these operations. Operating in the wavelet domain enables one to perform these operations progressively in a coarse-to-fine fashion, operate on different resolutions, manipulate features at different scales, and localize the operation in both the spatial and the frequency domains. Performing such operations in the wavelet domain and then reconstructing the result is also often more efficient than performing the same operation in the standard direct fashion. In this paper we demonstrate the applicability and advantages of this framework to three common types of image operations: image blending, 3D warping of images and sequences, and convolution of images and image sequences.

1 Introduction

Many common and useful operations on images in computer graphics and image processing are inherently parallel and can be expressed using linear combinations of matrices. Some of these matrices are related to the image being operated upon, while others represent the operation itself. Such operations include, for example, image blending, image convolution, and various kinds of image warping algorithms. This paper presents a new framework for representing and performing such operations in the wavelet domain. The basic idea is that given a linear combination of matrices, we compute the wavelet transform for each matrix and perform the linear combination on the wavelet coefficients. Applying the inverse wavelet transform to the resulting coefficients yields the desired linear combination.

Performing such operations in the wavelet domain is advantageous in several respects:

[*]idrori@alum.cs.huji.ac.il
[†]danix@cs.huji.ac.il

Sparse representation: With a suitable choice of a wavelet basis, the wavelet decomposition of natural images is typically much sparser (has fewer non-zero coefficients) than the original representation. The same is true for a certain class of operators (Calderon-Zygmund), and this property has been utilized to speed up various numerical operations [2, 3]. In our case, a sparse representation of images (as well as certain matrices arising from image warping equations) results in faster computation of their linear combinations, since we must combine only the unique non-zero coefficients.

Multi-resolution and progressive computation: The wavelet decomposition represents images and matrices at multiple scales. Such a representation makes it easy to perform operations at multiple resolutions, as well as in a progressive, coarse-to-fine fashion.

Space-frequency locality: The wavelet transform has nice localization properties in both the spatial and the frequency domains. Thus, by selectively operating on subsets of coefficients it is possible to restrict the effect of the operation both spatially and in frequency.

Compatibility with emerging standards: Over the past decade the wavelet transform has been recognized as a preferred tool for image and video compression and has been selected as the fundamental building block of the emerging JPEG-2000 standard. Thus, it is very likely that many of the images that we will be working with in the future will be represented in the wavelet domain to begin with. Wavelet domain operations will allow processing of such images without having to reconstruct them first.

In this paper we demonstrate these advantages by discussing in detail the application of our framework to three important types of operations on images: image blending, 3D warping for image-based rendering, and convolution. The wavelet-based 3D warping approach presented here is an improved version of an algorithm we described in a previous publication [15]. In particular, we present an improved tile-based memory management scheme, which makes the technique effective for images of unlimited size. We also present a more comprehensive discussion regarding the choice of wavelet bases, and describe in more detail the data structures used for effectively representing sparse wavelet decompositions. The parts of the paper concerned with image blending and convolution have not been published before.

1.1 Image-based rendering and 3D warping

Many image-based rendering algorithms use pre-rendered or pre-acquired *reference images* of a 3D scene in order to synthesize novel views of the scene. The central computational component of such algorithms is 3D image warping, which performs the mapping of pixels in the reference images to their coordinates in the target image. In Section 4 we present *wavelet warping* — a new class of forward 3D warping algorithms for image-based rendering. We rewrite the 3D warping equations as a pointwise quotient of linear combinations of matrices. Rather than computing these linear combinations in a standard manner, we first pre-compute the wavelet transforms of the participating matrices. Next, we perform the linear combinations using only the unique non-zero wavelet transform coefficients. Applying the inverse wavelet transform to the resulting coefficients yields the desired linear combinations.

We describe in detail wavelet warping algorithms for three common types of 3D image warps: planar-to-planar, cylindrical-to-planar, and spherical-to-planar. Cylindrical and spherical panoramas and movies are becoming increasingly common in application areas, such as entertainment, real estate, virtual tourism, and electronic retail. Current viewers allow the user to interactively change the viewing direction [6]. By using depth information, a 3D warper enables users to change the *viewing position* (center of projection), in addition to the viewing direction [24]. A fast 3D warper enables users to view a scene interactively. We will show that the wavelet warping algorithm is at least as fast as the most efficient warping algorithm known to date for planar and cylindrical warps, and is nearly twice as fast in the spherical case.

Perhaps more importantly, our wavelet warping algorithms support progressive multi-resolution rendering. For example, consider an object whose image-based model consists of one or more high-resolution reference images. The high resolution may be necessary for a close-up view of the object, but for most views of a 3D scene containing the object a much lower resolution suffices. Our approach makes it possible to perform the warp at the appropriate coarser resolution, without unnecessarily warping each and every pixel in the reference images. Multi-resolution warping can also be achieved within a standard warping framework by using an over-complete pyramid-based image representation (e.g. a Laplacian pyramid), but at a cost of increasing the size of the representation. Multi-resolution wavelet warping has the advantage that the computation is progressive: a low resolution result can be progressively refined without redundant computations.

In Section 4.7 we present a new algorithm for warping an entire sequence of images with depth to a novel view. This algorithm is also based on wavelet warping, and it utilizes the temporal coherence typically present in image sequences or panoramic movies to achieve considerable speedups over frame-by-frame warping.

1.2 Convolution

Convolution is probably the most widely used operation in image processing. *Wavelet convolution*, described in Section 5, is shown to be more efficient than standard convolution, even for small images and small kernels. Wavelet convolution also enables rapid convolution of image sequences with 3D filters, which is a useful operation for video processing.

In this application, we also show that by performing *lossy* wavelet compression accuracy can be gracefully traded off for speed, providing a way to obtain a visually accurate approximation to the result of the convolution very rapidly. Both exact and approximate wavelet convolution should prove very useful in image editing applications, since they result in faster response time, thus allowing users to preview the results of their operations interactively and at the full resolution of the image.

Similarly to wavelet warping, wavelet convolution can also be performed at different resolutions. Furthermore, by performing the operation on subsets of the wavelet coefficients, convolution can be restricted to operate only on a certain region in the image, or only on features at a certain scale, or both. This kind of flexibility is available neither in the standard convolution, nor in the FFT-based frequency domain convolution.

1.3 Related work

The idea of representing a scene as a set of reference images was introduced to computer graphics by Chen and Williams [7] and by McMillan and Bishop [24]. The equations of 3D warping are developed in detail in McMillan's PhD thesis [23]. Mark *et al.* [22] and Shade *et al.* [32] discuss different frameworks for image-based rendering and warping. Dally *et al.* [10] introduce the delta tree, a data structure that represents an object using a collection of images. They divide images into blocks and represent them in the frequency domain using the discrete cosine transform (DCT), but provide little detail regarding the warping of such images.

Smith and Rowe [34] address the issue of processing JPEG-compressed images in the compressed DCT domain. By performing pixel-wise and scalar addition and multiplication on JPEG-compressed images they are able to implement operations such as dissolving between two video sequences and video subtitling very efficiently (compared to uncompressing, processing, and compressing again). In a later paper [35] their methods are extended to the computation of arbitrary linear combinations of pixels in images of motion-JPEG video sequences. However, they do not address 3D warping of images and video sequences.

Their approach is tuned to the particularities of JPEG (block-based DCT, quantization, zig-zag scanning, etc.). The resulting algorithms are quite complicated. In contrast, our approach is applicable to any wavelet transform (although its effectiveness will depend on which transform is actually chosen), and results in very simple algorithms. Another difference between their approach and this work is in the goals. Their primary goal is to process compressed images (or video sequences) directly, without ever leaving the compressed domain. Our primary goal is to provide fast, progressive, and multi-resolution operations on ordinary data, by representing the data and/or the operation in the wavelet domain. Our approach is geared towards an interactive setting, where operations are performed in the wavelet domain, but the results are typically reconstructed right away for display.

Beylkin *et al.* [2, 3] describe fast computation of linear operators on arbitrary vectors. They apply the non-standard wavelet transform to the linear operator matrix, and to the input vector. Next, the matrix-vector product is computed in the wavelet domain, by summing the contributions from different scales. Finally, the result is obtained by reconstruction — applying the inverse transform to the product. Using their method it is possible to compute convolutions in the wavelet domain [14, 29]. However, in order to represent a convolution of an $n \times n$ image as a matrix-vector multiplication, the corresponding operator matrix must be $n^2 \times n^2$. Computing the nonstandard wavelet decomposition in this case takes $O(n^4)$. In contrast, our approach is to represent convolution as a linear combination of images, and to apply the wavelet decomposition to these images. Each decomposition takes $O(n^2)$ operations, i.e., time that is linear in the number of pixels.

2 Wavelets

Wavelets are a powerful mathematical tool for hierarchical multi-resolution analysis of functions. They have been effectively utilized in many diverse fields, including approximation theory, signal processing, physics, astronomy, and image processing [21]. Wavelets have also been applied to a wide variety of problems in computer graphics [36]. In this section we briefly review wavelet-related terminology and concepts that will be used later in the paper.

Lifting: The lifting scheme [37] is a method for constructing wavelets in the spatial domain. It consists of three steps: (i) splitting the data into two subsets; (ii) computing the wavelet coefficients as the failure to predict one subset based on the other (high pass); (iii) computing the scaling function coefficients by updating the remaining subset (low pass).

Any discrete wavelet transform can be factored into lifting steps [11], thus allowing: (i) in-place computation of the wavelet transform; (ii) faster computation, asymptotically reducing the complexity by a factor of four [30]; (iii) construction of wavelet transforms that map integers to integers [5].

Integer wavelets: Integer wavelet transforms operate on integer valued signals to produce integer valued wavelet coefficients. Integer wavelet transforms have been effectively used for lossless compression of images [5]. As presented in [5], the results of the invertible integer wavelet transforms are integer, while the computations are done with floating point numbers. In our implementation, all transforms are computed using integer arithmetic. Computations are done with integer numbers, using only addition, subtraction and shift operations.

Multiple dimensions: The 2D wavelet transform of a matrix or an image can be constructed using the 1D wavelet transform in two ways: the *standard decomposition* and the *nonstandard decomposition*. In this paper, we use the nonstandard decomposition, which is computed by applying the 1D transform alternating between rows and columns of the 2D matrix or image. The 3D wavelet transform of an image sequence can be constructed using the 1D transform in both the standard and non-standard forms, or by a combination of the 1D transform with the 2D transforms. In this paper we use the 1D transform in the temporal dimension and the 2D nonstandard transform in the spatial dimensions. The transform is applied recursively to the low frequency sub-bands. In the general case, the nonstandard wavelet transform is computed with less than $d/(1-2^{-d})kn^d$ operations, where d is the dimension, k the filter length, and n the number of samples in each dimension.

Compression: The wavelet transform can be used for lossless and lossy compression. Compression is achieved by transforming a signal using a wavelet which is adapted to it, and by truncating the resulting wavelet coefficients below a threshold. Lossy wavelet compression, for example, begins by computing the wavelet transform of a signal. Next, as many small wavelet coefficients as possible are zeroed out, so long as the overall error stays below a threshold ϵ in some L^p norm. Only the remaining nonzero coefficients are stored, resulting in a lossy compressed representation of the signal. A detailed exposition and error analysis of wavelet image compression is presented by DeVore *et al.* [13], where it is claimed that the appropriate error metric to use is L^1.

2.1 Linear combinations of matrices

Our approach is based on fast and progressive computation of linear combinations of matrices in the wavelet domain. Let $\mathbf{A}$ be a 2D matrix that can be expressed as a linear combination of matrices: $\mathbf{A} = \sum_i \alpha_i \mathbf{A}_i$, and let T be a 2D wavelet transform. Since T is an invertible linear operator, we can express $\mathbf{A}$ as

$$\mathbf{A} = T^{-1}\left(T\left(\mathbf{A}\right)\right) = T^{-1}\left(\sum_i \alpha_i T\left(\mathbf{A}_i\right)\right). \tag{1}$$

In other words, $\mathbf{A}$ can be computed in the wavelet domain, by computing the wavelet transform (decomposition) of each matrix $\mathbf{A}_i$, linearly combining the resulting wavelet coefficients, and applying the inverse transform (reconstruction). If the wavelet decompositions $T\left(\mathbf{A}_i\right)$ are sparse, this computation can be done rapidly by operating only on the nonzero coefficients of each transform.

The coefficients of a wavelet transform can be classified into several disjoint groups. The coefficients in each group correspond to a different resolution (scale). We utilize this property in order to compute linear combinations of matrices in a progressive, multi-resolution fashion. The coarsest level of the desired matrix $\mathbf{A}$ is obtained by computing the linear combination of the coarsest-level $T\left(\mathbf{A}_i\right)$ coefficients only, and reconstructing up to the corresponding level. In order to refine the result one level further, we compute the linear combination of the next finer level of $T\left(\mathbf{A}_i\right)$ coefficients, and perform one more level of reconstruction. This process may continue until the finest level has been processed. In this manner, intermediate results corresponding to different resolutions become available while the computation proceeds, without redundant computations.

2.2 Choice of Transform

There are two main requirements that a wavelet transform should satisfy in order to be suitable for our framework.

1. The transform should be sparse.
2. Reconstruction (inverse wavelet transform) should be fast to compute.

In practice, the first requirement implies that the wavelet basis should be adapted to the image operation in which it will be used. For example, in image blending and convolution it is important for the wavelet transform to be sparse when applied to the images being operated upon (typically natural images). In contrast, in wavelet warping we apply the transform to matrices derived from reformulating the warping operation. These matrices possess a special structure, which can be represented very sparsely in the wavelet domain, if a suitable wavelet basis is used.

When choosing a wavelet basis for *image compression* there is a trade-off between the number of vanishing moments and the size of the support [21]. On the one hand, more vanishing moments imply more near-zero coefficients in the transform over smooth regions of an image. However, the number of vanishing moments is proportional to the support size, and higher support size implies longer decomposition and reconstruction times, as well as large coefficients whenever the support overlaps a sharp transition in the image.

In order to choose a suitable transform for the applications described in this paper we have experimented with the following wavelet transforms:

- The S transform (sequential transform), which is an integer version of the Haar transform using lifting steps [5]. This transform has one vanishing moment.

- The S+P transform [31] (sequential transform + prediction), which has two vanishing moments.

- The TS transform [38], which is an integer version of the (3,1) biorthogonal wavelet transform of Cohen-Daubechies-Feauveau [8].

- A slightly modified version of the second order interpolating wavelet transform, $I(2,2)$ [37]. The modification consists of omitting the update phase of the lifting scheme. This transform has two vanishing moments.

- The popular commonly used 9-7 wavelet transform [8]. This transform has four vanishing moments.

To achieve faster reconstruction we choose transforms with smaller support size, and therefore fewer vanishing moments. This rules out the 9-7 transform which is considerably slower that the other transforms. In particular, this transform requires floating point arithmetic, whereas the other transforms can be implemented using only integer additions and shifts. The S+P and TS transforms are similar. They are both special cases of the same transform, which is factored into the S transform followed by an additional lifting step, but with different prediction coefficients. For our purposes it is sufficient to experiment with the more efficient TS transform.

In order to assess the speed and the sparsity of the remaining three wavelet bases (S, TS, and $I(2,2)$) we gathered the relevant statistics over a database of 1015 stock photography images. The subjects of these photographs are varied and include landscapes, buildings, people, products, and more. Nine randomly chosen images from this database are shown in Figure 1. Each image was transformed from RGB to YIQ color space and processed at full (640x384) and at half (320x192) resolutions. The results are summarized in Tables 1 and 2 and in the plots of Figure 2.

Our experiments indicate that all three transforms provide roughly the same sparsity of wavelet domain representation for natural images. Note that the percentage of remaining coefficients is typically higher when operating on the half-resolution versions of the image. Decreasing the resolution results in smaller smooth regions in the images, and applying a transform with few vanishing moments yields fewer near-zero coefficients over these regions.

In terms of speed, the S and $I(2,2)$ transforms are the fastest (the S transform is slightly faster), while the TS transform is slower by a factor of roughly 2. Consequently, the S transform was chosen for wavelet domain image blending (Section 3) and for wavelet

Figure 1: Nine images chosen at random from a database of 1015 stock photographs.

Transform:	S transform		TS transform		I(2,2) transform	
Resolution:	640x384	320x192	640x384	320x192	640x384	320x192
Rec. time:	12.8 (0.7)	3.2 (0.3)	27.7 (1.4)	6.9 (0.7)	13.4 (0.7)	3.3 (0.3)
% non-zero:	50.3 (13.7)	66.4 (12.5)	50 (12.8)	64.9 (12.3)	51.7 (11.2)	65.4 (10.9)

Table 1: A comparison between the S, TS, and I(2,2) transforms on a database of 1015 stock photography images. For each transform and each image resolution we list the mean reconstruction time in milliseconds and the mean percentage of remaining (non-zero) coefficients. The numbers in parentheses give the standard deviation corresponding to each mean.

Transform/Channel	Y	I	Q
none:	153	51	26
S:	134	36	19
TS:	121	32	18

Table 2: The average number of distinct non-zero values in a wavelet-transformed image for each of the Y, I, Q channels. The average was obtained over 102 randomly chosen images out of a database of 1015 stock photographs.

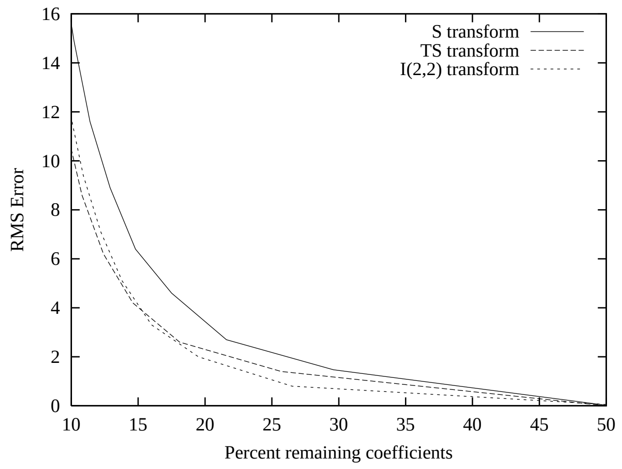


Figure 2: Lossy compression with the S, TS, and I(2,2) transforms: the RMS error of an image is plotted as a function of remaining non-zero coefficients. The data for this plot was obtained by averaging over a database of 1015 stock photography images.

domain convolution (Section 5). For wavelet warping (Section 4) we chose the $I(2,2)$ transform for reasons that will become apparent in Section 4.2.

So far we have only considered lossless wavelet domain representation of images (only coefficients that become identically zero as a result of the wavelet transform are eliminated from the representation). Lossy representations obtained by zeroing out small wavelet coefficients yield a drastic reduction in the number of remaining coefficient in return for a modest increase in RMS error, as demonstrated by the plots in Figure 2. Such representations can be acceptable if numerical accuracy is not critical, as will be demonstrated in Section 5.4. When choosing a wavelet transform for a lossy wavelet domain representation one additional requirement must be taken into account: graceful degradation in visual quality of the image. In this respect we found the slower biorthogonal TS transform to be superior to the S transform. More specifically, the lossy TS transform tends to produce smoother and more visually accurate results compared to the lossy S transform, which introduces blocky artifacts.

2.3 Data Structures

In order to take advantage of the sparsity of wavelet-transformed images and matrices a suitable data structure is required. We designed a data structure in which we represent separately three different group of coefficients. The first group consists of the coeffcents belonging to the two finest levels in the wavelet decomposition, accounting for 15/16 of the coefficients. Many of these coefficients are zero. Furthermore, the distinct values of the non-zero coefficients typically form a set of an even smaller size (see Table 2). In order to take advantage of these two properties we store all of the distinct non-zero coefficient values in a lookup table. The non-zero coefficients themselves are stored in an array. Each array entry contains the position of a non-zero coefficient in the decomposition, along with an index into the lookup table. This representation allows us to perform linear combinations very efficiently, as each distinct non-zero value must only be multiplied once. It should be noted that in order to perform fair comparisons between wavelet-domain and standard operations we used look-up tables of pre-multiplied wavelet coefficients only when the corresponding standard algorithm also uses look-up tables of premultiplied values.

The two remaining groups of coefficients are represented as follows. The the two coarsest levels in the decomposition, which contain very few coefficients to begin with, are represented in a straightforward manner using a simple array containing the values of the coefficients. Finally, the remaining (intermediate) levels are simply represented by arrays containing in each entry the position and value of a single non-zero coefficient.

3 Blending of multiple images

In this section we demonstrate a simple yet useful application of our approach: interactive re-rendering of a scene under novel illumination by fast blending of images acquired under varying illumination conditions.

The additive nature of illumination has been exploited by several researchers for efficiently re-rendering a scene by linearly combining a collection of images. For example, Nimeroff *et al.*[25] precompute a collection of basis images for an architectural scene from which the illumination of the scene under different natural illumination conditions can be reconstructed. Debevec *et al.*[12] describe a lighting rig for acquiring the reflectance field of a human face. The reflectance field is represented as a two-dimensional array of 2048 images, each corresponding to a different illuminant direction. Linearly combining these images with different weights yields the appearance of the face under arbitrary new illumination conditions. The linear combination is computed using directly the discrete cosine transform coefficients of the images [34]. In order to perform the computation progressively from coarse to fine resolution an over-complete representation of the reflectance field is used, storing compressed reflectance maps at full, half and quarter resolutions.

Recently, several researchers [1, 28] have independently shown that irradiance environment maps (for rendering Lambertian objects under distant illumination) can be very well approximated by a nine-dimensional linear subspace. Thus, from any given view, the appearance of an arbitrary Lambertian shape under arbitrarily distributed distant illumination can be reproduced by linearly combining nine images.

All of the applications described above could benefit from the ability to perform fast progressive blending of multiple images. Fortunately, this is easily done in our framework via a straightforward application of equation (1): the matrices $\mathbf{A}_i$ are simply the images to be blended. In order to illustrate wavelet-domain multiple image blending we use a set of ten 352×512 wavelet-transformed images of a face under varying illumination conditions (from [12]) to interactively re-render the same face image under novel illumination. Figure 3 shows four of these face images.

The ten images were pre-processed by applying the S transform in place. As predicted

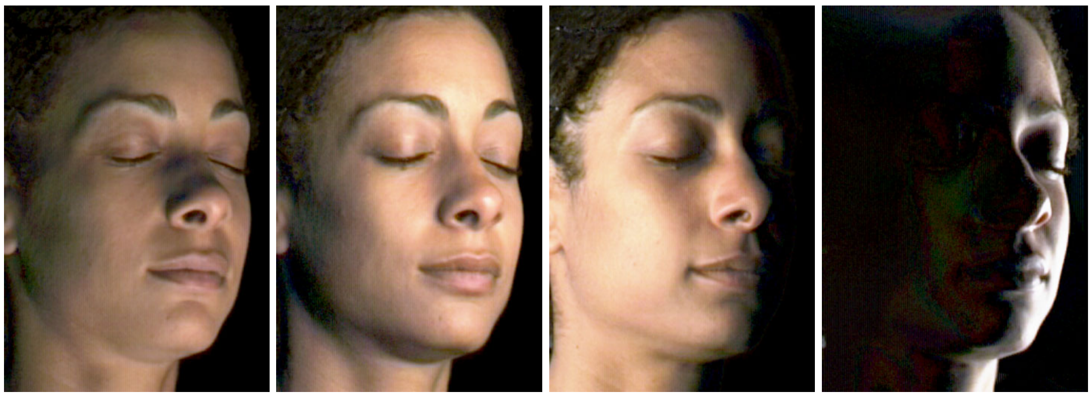
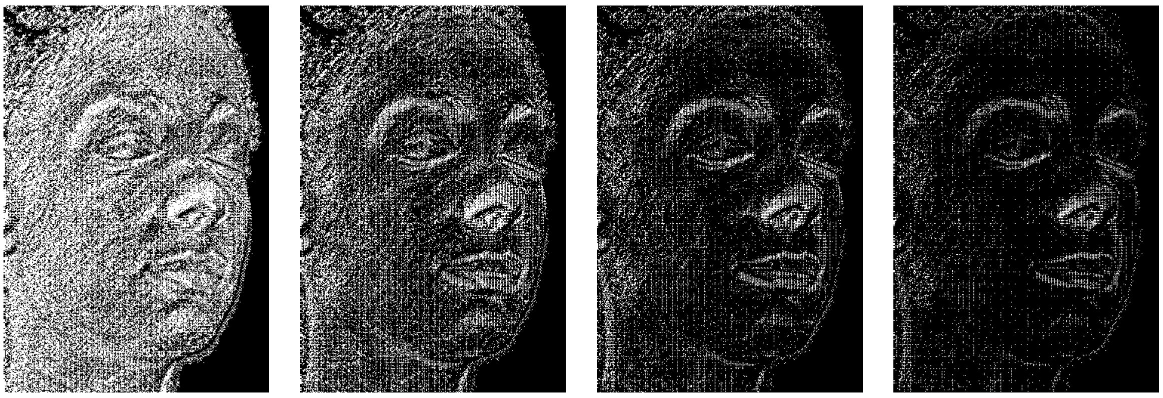


Figure 3: Images of a human face under different illumination conditions.

by our experiments in Section 2.2, the resulting lossless wavelet domain representation is rather sparse. Figure 4 visualizes the wavelet domain representation of the leftmost image of Figure 3. All of the non-zero wavelet coefficients in this representation (regardless of their magnitude) are shown as white pixels. In a lossless wavelet-domain representation for this image only half of the coefficients are non-zero. This number can be further drastically decreased if some loss of accuracy is tolerable.



threshold: 1/255	3/255	5/255	7/255
% non-zero: 50.2	23.6	11.0	5.4

Figure 4: The wavelet-domain representation of the leftmost image of Figure 3. White pixels indicate non-zero wavelet coefficients. The leftmost image corresponds to a lossless wavelet transform. The remaining three images on the right show the result of lossy transforms, with the threshold increasing from left to right.

The image blending process begins by computing the linear combination of the coarse levels non-zero wavelet coefficients, progressively proceeding to finer levels. Intermediate results are incrementally reconstructed and displayed. This process is illustrated in Figure 5, which shows the blended result at three different resolutions. The quarter- and half-resolution images are scaled to the full size by using linear interpolation available on virtually all of today's commodity graphics accelerators. As can be observed, the differences between the different resolutions are hard to perceive (mostly it can be seen that the lower resolution images are slightly blurrier).

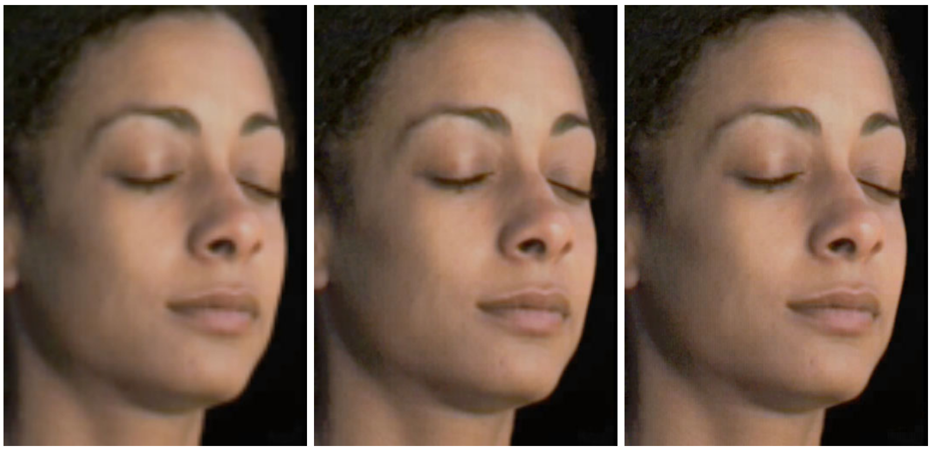


Figure 5: The result of the blend at quarter resolution (left), half resolution (center), and full resolution (right).

The time to blend the ten 352×512 images in the standard manner takes 278.3 ms (milliseconds) in our Java implementation, whereas progressive wavelet-domain blending of the same images yields a quarter-resolution result after 10.6 ms, a half-resolution result after 44.5 ms, and the final full-resolution result after 174.1 ms. These times include both the time to compute the linear combinations of wavelet coefficients and the reconstruction time (in order to obtain a displayable image). Thus, our approach provides a non-redundant progressive multi-resolution algorithm for blending multiple images without sacrificing computation speed. In order to maintain interactive rates with a much larger number number of blended images (such as 2048), lossy wavelet-domain representation can be used, trading speed for image quality.

4 3D Warping

This section describes the application of our approach to various 3D warping mappings, which are in the core of most image-based rendering algorithms. We begin by briefly reviewing the 3D warping equations (the reader is referred to McMillan's PhD thesis [23] for detailed derivations). Following the review we show how to express (parts of) the warping operation as a linear combination of matrices, thereby paving the way for *wavelet domain 3D warping*.

Most image-based rendering algorithms use *forward* 3D warping, which maps pixels from a reference (source) image to the desired (target) image, according to the following general equation [23]:

$$\mathbf{p}_2 = \mathbf{M}_2^{-1} \left(\delta(\mathbf{p}_1) \left(\mathbf{o}_1 - \mathbf{o}_2\right) + \mathbf{M}_1 \left(\mathbf{p}_1\right)\right) .$$

The 3-vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are the homogeneous image coordinates of the source and target pixels, respectively. The matrices $\mathbf{M}_i$ map pixel coordinates to 3D rays, and the points $\mathbf{o}_i$ are the centers of projection. The generalized disparity $\delta(\mathbf{p})$ is inversely proportional to the depth at pixel $\mathbf{p}$.

More specifically, the forward mapping from reference image space coordinates (x, y) to target image space coordinates (u, v) is expressed as:

$$u = \frac{f_1(x,y)}{f_3(x,y)} \qquad v = \frac{f_2(x,y)}{f_3(x,y)} \tag{2}$$

where the functions $f_i(x, y)$ depend on the type of warp. For example, when warping a planar reference image to a planar target image, $f_i(x, y)$ can be expressed as:

$$f_i(x,y) = \begin{bmatrix} p_{i1} & p_{i2} & p_{i3} & p_{i4} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \\ \delta(x,y) \end{bmatrix} \tag{3}$$

The scalars p_{ij} are dependent upon the view matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ and the vector $\mathbf{o}_1 - \mathbf{o}_2$ between the two centers of projection. The result is valid when $f_3(x, y) > 0$ (point is in front of camera), and when (u, v) are in the range of target image space coordinates.

When warping a cylindrical reference image to a planar target image, the equations become:

$$f_i(x,y) = \begin{bmatrix} c_{i1} & c_{i2} & c_{i3} & c_{i4} \end{bmatrix} \begin{bmatrix} \sin(2\pi x/w) \\ \cos(2\pi x/w) \\ y_0 + (y_1 - y_0)y/h \\ \delta(x,y) \end{bmatrix} \tag{4}$$

where w and h are the width and height of the image in pixels, and y_0 and y_1 define the vertical field-of-view of the cylindrical image. Similarly, a warp from spherical image coordinates to a planar image is defined by:

$$f_i(x,y) = \begin{bmatrix} s_{i1} & s_{i2} & s_{i3} & s_{i4} \end{bmatrix} \begin{bmatrix} \sin(2\pi x/w)\sin(\pi y/h) \\ \cos(2\pi x/w)\sin(\pi y/h) \\ \cos(\pi y/h) \\ \delta(x,y) \end{bmatrix} \tag{5}$$

4.1 Wavelet warping

In order to perform 3D warping in the wavelet domain, we express the warping equations as element-wise divisions between linear combinations of four matrices. Let $\mathbf{F}_i$ denote the matrix of all the values $f_i(x, y)$, and let $\mathbf{U}$ and $\mathbf{V}$ denote the matrices containing all of the warped u and v target coordinates. Using these matrices we rewrite equation (2) as

$$\mathbf{U} = \frac{\mathbf{F}_1}{\mathbf{F}_3} \qquad \mathbf{V} = \frac{\mathbf{F}_2}{\mathbf{F}_3},$$

where

$$\mathbf{F}_i = m_{i1}\mathbf{A} + m_{i2}\mathbf{B} + m_{i3}\mathbf{C} + m_{i4}\mathbf{D}. \tag{6}$$

In the planar-to-planar warp, for example, the linear combination coefficients m_{ij} are the p_{ij}'s from equation (3), and the matrices are defined as follows:

$$\mathbf{A} = [x]_{x,y} \quad \mathbf{B} = [y]_{x,y} \quad \mathbf{C} = [1]_{x,y} \quad \mathbf{D} = [\delta(x,y)]_{x,y} \tag{7}$$

Thus, the matrix **A** is simply a linear ramp, increasing from left to right; all of its rows are the same vector $[0, 1, \ldots, n-1]$. Similarly, the matrix **B** is a linear ramp, and all of its columns are the same vector. The matrix **C** is constant. The wavelet transform of these matrices is extremely sparse, and the efficiency of our wavelet warping algorithm stems from this sparse representation.

In the cylindrical-to-planar case the matrices are slightly more complicated:

$$\begin{array}{ll} \mathbf{A} = \left[\sin(2\pi x/w)\right]_{x,y} & \mathbf{B} = \left[\cos(2\pi x/w)\right]_{x,y} \\ \mathbf{C} = \left[y_0 + (y_1 - y_0)y/h\right]_{x,y} & \mathbf{D} = [\delta(x,y)]_{x,y} \end{array} \tag{8}$$

Still, note that each of the matrices **A** and **B** is a function of a single variable x, which means that in each of these two matrices all of the rows are equal. Similarly, **C** is a function of y, and therefore all of the columns are equal. Both the standard cylindrical-to-planar warp and our wavelet warping algorithm exploit this structure to save computations.

Finally, in the spherical-to-planar case the matrices are:

$$\begin{array}{ll} \mathbf{A} = \left[\sin(2\pi x/w)\sin(\pi y/h)\right]_{x,y} & \mathbf{B} = \left[\cos(2\pi x/w)\sin(\pi y/h)\right]_{x,y} \\ \mathbf{C} = \left[\cos(\pi y/h)\right]_{x,y} & \mathbf{D} = [\delta(x,y)]_{x,y} \end{array} \tag{9}$$

In this case only **C** is a function of a single variable y, and therefore all of its columns are equal.

The wavelet warping operation consists of three steps: (i) computation of linear combinations (equation (6)), (ii) reconstruction, and (iii) clipping and element-wise divide. The first step is carried out in the wavelet domain, as illustrated in Figure 6. Thus, following equation (1), we compute the matrices $\mathbf{F}_i$ as follows:

$$\begin{aligned} \mathbf{F}_i &= T^{-1}T\left(m_{i1}\mathbf{A} + m_{i2}\mathbf{B} + m_{i3}\mathbf{C} + m_{i4}\mathbf{D}\right) \\ &= T^{-1}\left(m_{i1}T(\mathbf{A}) + m_{i2}T(\mathbf{B}) + m_{i3}T(\mathbf{C}) + m_{i4}T(\mathbf{D})\right) \end{aligned} \tag{10}$$

Several things should be noted at this point:

- The matrices **A**, **B**, and **C** depend only on the type of warp (planar, cylindrical, or spherical), and are independent of the reference or the target images. Consequently, $T(\mathbf{A})$, $T(\mathbf{B})$, $T(\mathbf{C})$ are precomputed once for each type of warp, and then reused for all warping operations.
- The matrix **D**, which is the disparity image of the reference view, is independent of the target view, and $T(\mathbf{D})$ is precomputed once for each reference view.
- The scalars m_{ij} are dependent upon both the reference and the target views, and are calculated once for each target view, the same as in a standard warp.
- The disparity values in **D**, as well as the entries of **A**, **B**, and **C** (in the cylindrical and spherical cases), contain floating point values. These values are first mapped into an appropriate integer range, since our implementation uses an integer wavelet transform.

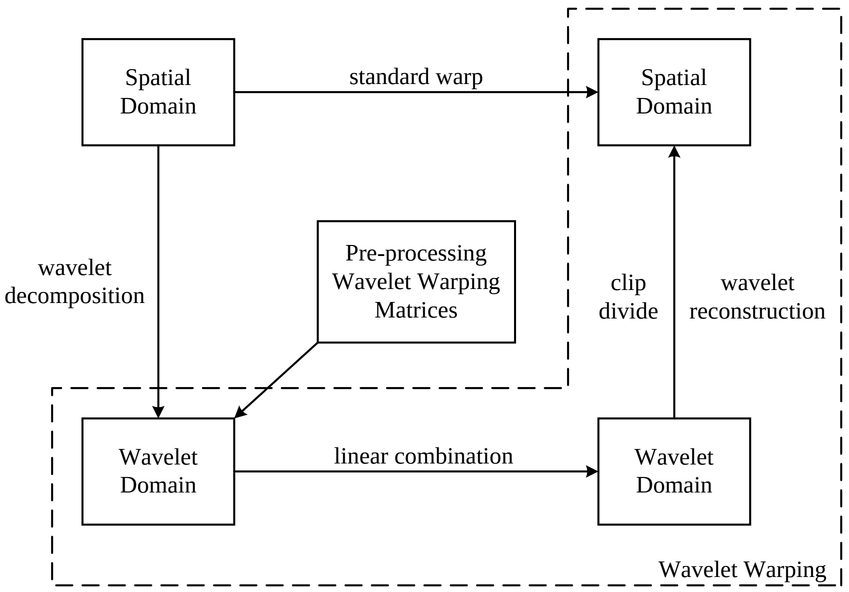


Figure 6: Standard warp vs. wavelet warp.

4.2 Choice of wavelet transform

As discussed in Section 2.2, there are two requirements that a suitable wavelet transform should satisfy: (i) the transforms $T(\mathbf{A})$, $T(\mathbf{B})$, $T(\mathbf{C})$, and $T(\mathbf{D})$ should be sparse; (ii) the reconstructions (inverse wavelet transforms) should be fast to compute. Based on the experiments reported in Section 2.2 we chose a slightly modified version of the second-order interpolating wavelet transform, $I(2,2)$ [37]. The modification consists of omitting the update phase of the lifting scheme. The resulting transform requires $\frac{8}{3}n^2$ operations to decompose an $n \times n$ matrix using the 2D nonstandard wavelet transform.

The wavelet coefficients of this transform measure the extent to which the original signal fails to be linear. In the case of a planar warp, the matrices $\mathbf{A}$ and $\mathbf{B}$ are simply linear ramps and matrix $\mathbf{C}$ is constant (eq. 7)). Consequently, the transforms $T(\mathbf{A})$ and $T(\mathbf{B})$ consist of two non-zero coefficients each, and $T(\mathbf{C})$ consists of a single non-zero coefficient. Note that this is *lossless compression* of the three matrices — they can be reconstructed exactly from these sparse transforms.

In the case of a cylindrical warp (eq. (8)) the transforms $T(\mathbf{A})$ and $T(\mathbf{B})$ have fewer than $\frac{1}{9}n^2$ nonzero coefficients each, while $T(\mathbf{C})$ has two non-zero coefficients. In the case of a spherical warp (eq. (9)) the transforms $T(\mathbf{A})$, $T(\mathbf{B})$ and $T(\mathbf{C})$ have fewer than $\frac{1}{9}n^2$ non-zero coefficients each. Once again, the compression of the matrices is lossless.

As for the disparities matrix $\mathbf{D}$, the number of non-zero coefficients depends, of course, on the reference image. In our experiments, roughly one third of $T(\mathbf{D})$ coefficients were non-zero. Although the number of non-zero coefficients can be decreased further by lossy wavelet compression, it is not beneficial to do so. As we shall see in the next section, the computational bottleneck of wavelet warping lies in the reconstruction stage. A slight reduction in the number of coefficients does not significantly improve performance, while a more drastic truncation causes errors in the mapping, resulting in visible artifacts.

4.3 Analysis and comparison

The complexity of 3D warping is linear in the number of warped pixels, with the constant factor depending on the type of warp. In the standard warping algorithm, a planar-to-planar warp computes equation (3) using an incremental loop, which requires a single addition for each increment in x or y, plus an additional multiplication and addition for the generalized disparity term for each f_i [32]. The clipping is followed by two divisions. Thus, the total number of operations to warp an $n \times n$ image is $11n^2$.

In the cases of cylindrical-to-planar and spherical-to-planar warps, efficient computation requires that the terms in matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ be precomputed and stored in lookup tables (LUT). In the cylindrical-to-planar case, as pointed out earlier, there are only n distinct terms in each of the three matrices. Thus, each m_{ij} must be multiplied with only n, rather than n^2 different terms. A similar optimization is performed in the spherical-to-planar case when computing $m_{i3}\mathbf{C}$.

In contrast, since the wavelet warping algorithm performs operations only on the non-zero elements of the transformed matrices, the total number of operations required to compute equation (10) is $t - |T(\mathbf{D})|$ additions and $\hat{t}$ multiplications (where t is the total number of nonzero wavelet coefficients, and $\hat{t}$ the number of unique non-zero coefficients), plus the cost of the reconstruction step that takes $\frac{8}{3}n^2$ operations in our implementation. We perform three reconstruction steps (one for each $\mathbf{F}_i$), followed by clipping and two divisions per pixel. The results of the analysis are summarized in Table 3.

Operation	any wavelet warp	planar	cylindrical	spherical
Addition	$8n^2 + 3\,(t - \|T(\mathbf{D})\|)$	$6n^2 + 3n$	$6n^2 + 3n$	$9n^2$
Multiplication	$3\hat{t}$	$3n^2$	$3n^2 + 9n$	$9n^2 + 3n$
Division	$2n^2$	$2n^2$	$2n^2$	$2n^2$
Total	$10n^2 + 3\,(t - \|T(\mathbf{D})\|) + 3\hat{t}$	$11n^2 + 3n$	$11n^2 + 12n$	$20n^2 + 3n$

Table 3: Number of arithmetic operations in different types of warps. The columns labeled planar, cylindrical, and spherical refer to an efficient incremental implementation of the standard warp in each of these three cases.

As explained in Section 4.2, our choice of wavelet basis results in a total number of $t = |T(\mathbf{D})| + 5$ nonzero coefficients in the wavelet planar warp, $t < \frac{2}{9}n^2 + |T(\mathbf{D})| + 2$ in the cylindrical case, and $t = \frac{1}{3}n^2 + |T(\mathbf{D})|$ in the spherical case. The conclusion from our analysis is therefore that wavelet warping is as fast as the most efficient standard warp in the planar and cylindrical cases, while in the spherical warp we can expect speedups by factors between 1.54 and 2.

4.4 Memory management

The standard warping algorithm warps the reference image pixels to their destination, one by one, so its intermediate storage requirements are $O(1)$. In contrast, the wavelet warping algorithm described above must reconstruct the three matrices $\mathbf{F}_i$ before computing the target pixel coordinates. Thus, it requires $O(n^2)$ intermediate storage. When the warped image is sufficiently large, the intermediate storage requirements exceed the size of the L2 memory cache, resulting in a performance penalty. In this case, we split the matrices into the largest tiles of constant size that fit in the cache. The linear combination of each set of corresponding tiles is computed, and the reconstruction step is performed one tile at a time. The required intermediate storage is thus reduced to the size of a tile. In addition to avoiding cache misses, this modification also allows us to take advantage of the fact that many of the matrices consist of tiles that are identical, symmetric, or a rotation of one another.

The standard warping algorithm requires the storage of the reference image, as is. In contrast, in the wavelet warping algorithm the wavelet-transformed disparities and color matrices are represented using the data structures described in Section 2.3. Our experiments indicate that the total memory footprint was reduced to less than one third of that of the standard algorithm.

4.5 Empirical results

The theoretical analysis presented above has been validated experimentally. We have implemented our wavelet warping algorithm, as well as the standard warps: incremental planar-to-planar, LUT-based cylindrical-to-planar and spherical-to-planar, with the optimizations mentioned earlier. The algorithms were implemented in Java. All of the results reported in this paper were measured on a 450 MHz Pentium II processor. In all our comparisons we measured the entire warping time at full resolution, including reconstruction, clipping, and the divisions by the homogeneous coordinate. The averaged performance of the different warping algorithms (in frames per second) is summarized in Table 4.

Type of warp (# pixels warped)	standard warp	wavelet warp
planar (512×512)	6.5	7
cylindrical (512×256)	12	15
spherical (512×256)	7.7	14

Table 4: Measured performance (frames per second) of standard vs. wavelet warp in the planar, cylindrical, and spherical cases.

As predicted by our analysis, we found wavelet warping to be roughly as fast as the standard algorithm in the planar case and slightly faster (up to 25 percent) in the cylindrical case. Note that in the planar case the reference image has twice as many pixels as in the cylindrical case. This is the reason that the number of warps per second in the first row of the table is smaller almost by a factor of two. As expected, in the spherical case,

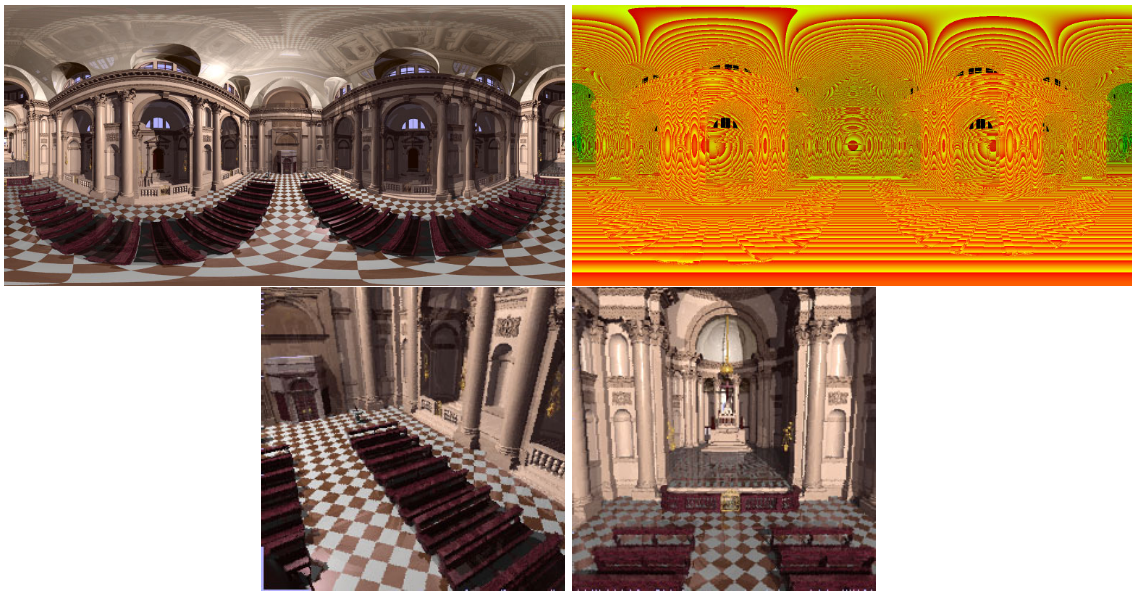


Figure 7: Spherical-to-planar image warping example.

wavelet warping outperforms the standard algorithm by a factor of roughly 1.8. Figure 7 shows a spherical-to-planar warping example. The two rectangular images in the top row show a spherical reference image with the corresponding depth image (depth values are 16-bit integers encoded using the red and green color channels). The lower left image is a novel planar view from a displaced position. Since only one reference image was used, some disocclusion artifacts can be seen. The lower right image is another planar view, taken from the original view point.

4.6 Multi-resolution wavelet warping

Suppose that we have an image-based representation of a 3D scene or object and would like to generate a novel view in which the scene or object is farther away from the viewpoint, and thus appears much smaller than in its reference views. Or perhaps, we are interested in rapidly generating lower resolution novel views while the view is changing, and then refine it if the camera stops moving. Our wavelet warping algorithms are well suited for such multi-resolution and/or progressive rendering.

When wavelet-warping a reference image to a target image of lower resolution there is no need to compute the entire linear combination (10) or perform a full reconstruction of the result. As explained in Section 2.1 it is possible instead to proceed from the coarsest to the finest scale, obtaining an intermediate result for any desired scale along the way. For example, if the target image resolution is twice smaller in each dimension, we stop before processing the wavelet coefficients of the finest level. As a result, computation is faster by a factor of four, and there are also four times fewer clip-and-divide operations. The warped coordinates (u, v) are still generated in the original range, so they are shifted right by one bit.

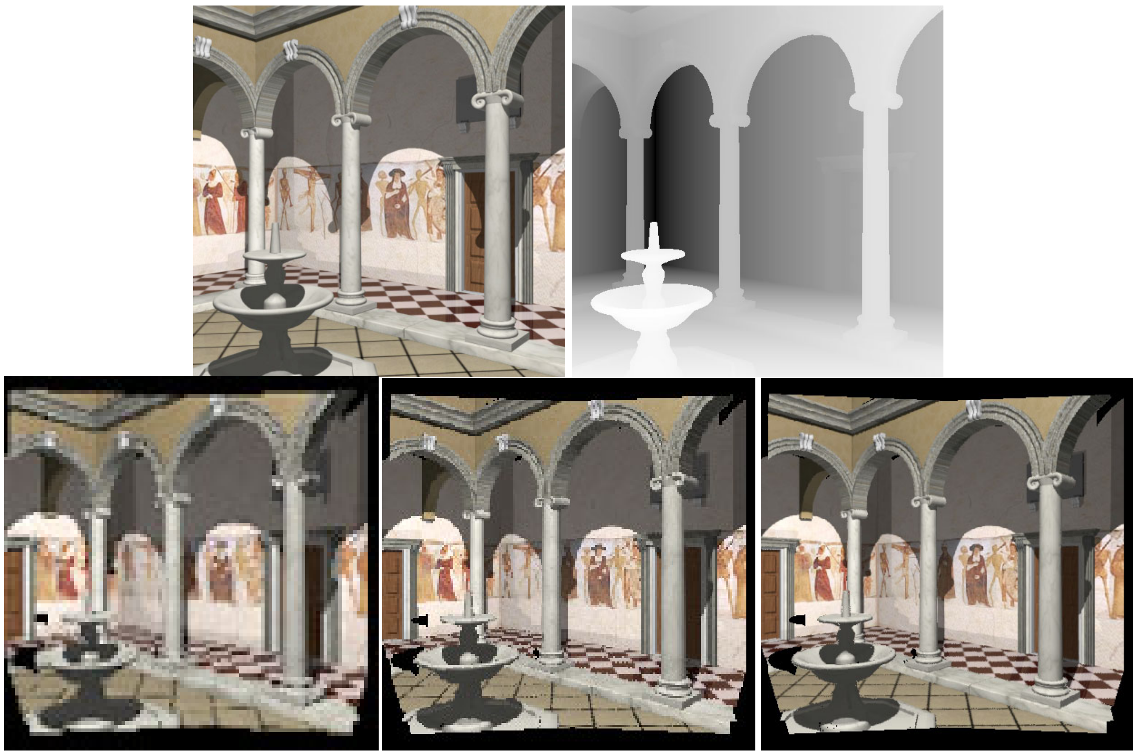


Figure 8: Multi-resolution planar wavelet warping.

When the target image resolution is lower, we must low-pass filter the color values of the reference image pixels, before copying them to the target image. Therefore, the color channels of the reference image are also represented in the wavelet domain. Prefiltered color values are obtained by reconstructing the image incrementally, as the resolution of the result is progressively refined from coarse to fine.

Figure 8 shows the results of planar-to-planar wavelet warping to different target resolutions. Two 512^2 reference views of a synthetic scene (with depth for each pixel) were warped to a common novel view. One of the reference views used and its corresponding depth image is shown in the top row. In the bottom row we show the target image generated at quarter (128^2), half (256^2), and full (512^2) resolutions, from left to right. In order to make the differences visible, all three images are shown at the same size in the figure.

When the target image is generated at full resolution, the wavelet warp takes roughly the same amount of time to compute as the standard incremental warp (around 7 fps). However, when the target resolution is reduced by half, wavelet warping becomes more than 4 times faster compared to full resolution warping (around 27 fps). At quarter resolution, wavelet warping is more than 16 times faster. Furthermore, wavelet warping has the advantage that the computation is progressive: a low resolution result can be progressively refined, simply by combining and reconstructing additional levels. It should be noted that progressive multi-resolution warping can also be achieved within a standard warping framework by using an *over-complete* pyramid-based image representation such as a the Laplacian pyramid [4, 27], but at the cost of increasing the size of the representation by a factor of $\frac{4}{3}$.

4.7 Temporal coherence

When warping an entire sequence of reference images taken using fixed viewing parameters (for example, a sequence that captures a dynamic event as seen from a particular viewpoint), temporal coherence can be utilized to make the computation faster than warping each frame individually. This is particularly easy to see using the matrix notation introduced earlier. The matrices **A**, **B**, and **C** are the same for any reference image, and the only matrix that differs between successive frames is the disparities matrix **D**. Let $\mathbf{D}^{(t)}$ denote the disparity matrix of frame t. Each frame of the warped sequence can be computed incrementally as

$$\mathbf{F}_i^{(t+1)} = \mathbf{F}_i^{(t)} + m_{i4}\Delta\mathbf{D}^{(t)}, \qquad \text{where} \quad \Delta\mathbf{D}^{(t)} = \mathbf{D}^{(t+1)} - \mathbf{D}^{(t)}.$$

Thus, if the difference matrices $\Delta\mathbf{D}^{(t)}$ are precomputed, it takes only $3kn^2$ additions, $3kn^2$ multiplications, and $2kn^2$ divisions to forward-warp an $n \times n \times k$ image sequence (saving $3kn^2$ additions compared to warping each frame individually). This is a simple observation, and the improvement is applicable to standard warping, but we have not encountered it in the image-based rendering literature.

Temporal coherence in this case is easily exploited in the context of wavelet warping. We precompute the 2D wavelet transforms $T(\Delta\mathbf{D}^{(t)})$. Each warped frame is then generated as follows:

$$\mathbf{F}_i^{(t+1)} = \mathbf{F}_i^{(t)} + T^{-1}\left(m_{i4}T(\Delta\mathbf{D}^{(t)})\right).$$

In other words, the differences between successive disparity images are multiplied by m_{i4} in the wavelet domain. Since the disparity of many pixels remains unchanged between consecutive frames, the wavelet transform of the differences is very sparse.

Utilization of temporal coherence in wavelet warping is demonstrated in the following “virtual studio” example. In this example we generate a new image sequence by warping images from three different sources (shown in the top row of Figure 9) into a common target view. The first source is a synthetic animation of a coffee-table following a circular trajectory. The second source is a still image of a synthetic 3D scene (a room). The third source is a video sequence of an actor performing in front of a Zcam — a real-time depth-sensing camera [39]. The target view is different from each of the original views of the three image sources. The three sources are wavelet-warped to the target view, where they are combined using a Z-buffer. The result is a video sequence where the actor is looking at the coffee-table that flies in a circle about him. Three frames from this sequence are shown in the bottom row of Figure 9. Using wavelet warping, as described earlier in this section, the target sequence is generated in real-time at 15 fps, which is faster by a factor of 2.5 than generating it by wavelet warping each frame individually.

Notice that the speedup when warping an image sequence is due to both the temporal coherence in the viewing parameters, i.e. the incremental formulation of the warping operation, and to the temporal coherence in disparity values, as many pixel depths remain unchanged between consecutive frames.

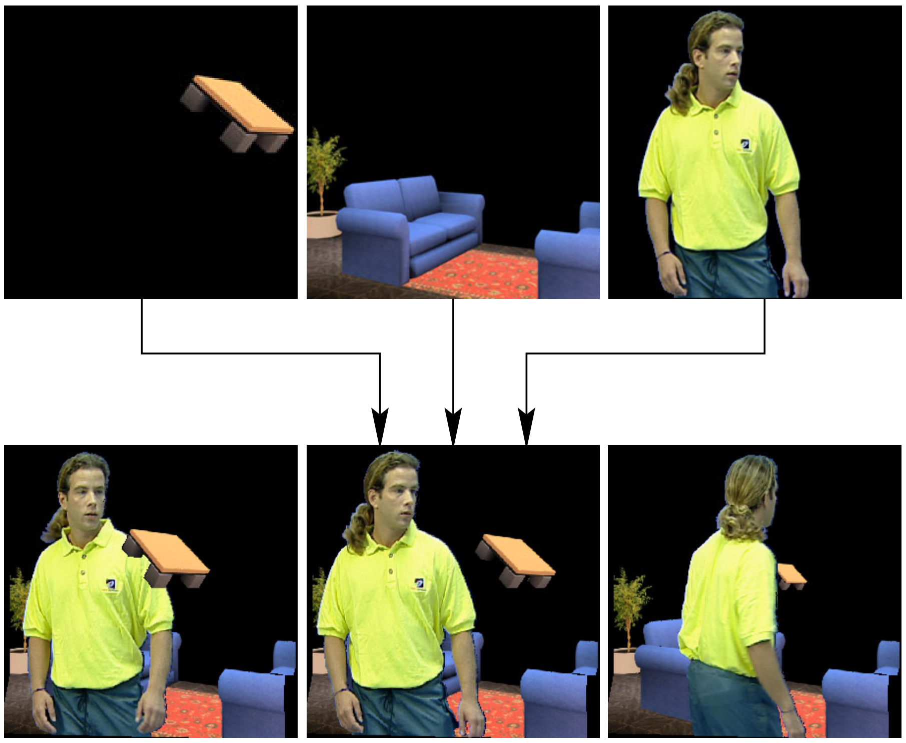


Figure 9: Real-time wavelet warping with temporal coherence.

5 Convolution

This section demonstrates the application of our approach to 2D and 3D convolution operation on images and image sequences, respectively. We begin with 2D image convolution. The *direct* convolution of an $n \times n$ image $\mathbf{A}$ with a $k \times k$ convolution kernel $\mathbf{K}$ is defined as

$$(\mathbf{K} * \mathbf{A})(u, v) = \sum_{i,j \in [0,k-1]} \mathbf{K}(i, j)\, \mathbf{A}(u - i, v - j).$$

The *Convolution Theorem* states that convolution can also be carried out in the frequency domain:

$$\mathbf{K} * \mathbf{A} = F^{-1}\left(F(\mathbf{K}) \cdot F(\mathbf{A})\right),$$

where F is the discrete Fourier transform, and $\cdot$ denotes componentwise multiplication.

5.1 Wavelet convolution

Let $\mathbf{A}^{ij}$ denote a shift of $\mathbf{A}$ by i, j in the X, Y directions. Since the convolution operator is commutative and distributive it is expressed as a weighted sum of k^2 matrices:

$$\mathbf{K} * \mathbf{A} = \sum_{i,j \in [-l,l]} k_{ij}\, \mathbf{A}^{ij}, \qquad \text{where} \quad l = \lfloor k/2 \rfloor$$

As before, this weighted sum is computed in the wavelet domain:

$$\mathbf{K} * \mathbf{A} \;=\; T^{-1}T\left(\sum_{i,j} k_{ij}\,\mathbf{A}^{ij}\right) \;=\; T^{-1}\left(\sum_{i,j} k_{ij}\,T(\mathbf{A}^{ij})\right). \tag{11}$$

Thus, after computing the k^2 wavelet transforms $T(\mathbf{A}^{ij})$, convolution is done by computing multiplication and additions with the nonzero wavelet coefficients. The result is the wavelet transform of the convolved image. In order to display the resulting image, wavelet convolution ends with a single reconstruction step.

Wavelet convolution of an image sequence with a 2D kernel is a simple extension. Since the convolution operator is distributive, we compute the convolution by taking the wavelet transform of the difference between consecutive images. The sequence can also be wavelet-convolved with a 3D kernel. In this case, we should precompute the wavelet transforms not only for shifts in X and Y, but also for shifts along the temporal dimension.

5.2 Symmetric wavelet convolution

A more efficient formulation is obtained by reducing in half the number of matrices involved in wavelet convolution. The weighted sum is split into two terms: the image multiplied by the kernel sum, and a weighted sum of the differences between the shifts $\mathbf{A}^{ij}$ and the image $\mathbf{A}$:

$$\mathbf{K} * \mathbf{A} = \left(\sum_{i,j\in[-l,l]} k_{ij}\right)\mathbf{A} + \sum_{(i,j)\neq(0,0)} k_{ij}\Delta\mathbf{A}^{ij}, \qquad \text{where} \quad \Delta\mathbf{A}^{ij} = \mathbf{A}^{ij} - \mathbf{A}$$

Note that the differences $\Delta\mathbf{A}^{ij}$ are even for i and j; in other words, it holds that $\Delta\mathbf{A}^{ij}(x,y) = -\Delta\mathbf{A}^{-i,-j}(x+i,y+j)$. Thus, the number of unique matrices in the sum is reduced from k^2 to $\lfloor\frac{k^2}{2}\rfloor + 1$. Utilizing this observation, wavelet convolution can be expressed as:

$$\begin{aligned}
\mathbf{K} * \mathbf{A}(x,y) &= \left(\sum_{i,j\in[-l,l]} k_{ij}\right)\mathbf{A}(x,y) + \sum_{i,j\in[-l,0]}\Big(k_{ij}\Delta\mathbf{A}^{ij}(x,y) - k_{-i,-j}\Delta\mathbf{A}^{ij}(x+i,y+j)\Big)\\
&= T^{-1}\left(\left(\sum_{i,j\in[-l,l]} k_{ij}\right)T(\mathbf{A})\right)(x,y)\\
&\quad + \sum_{i,j\in[-l,0]}\Big(k_{ij}\Delta\mathbf{A}^{ij}(x,y) - k_{-i,-j}\Delta\mathbf{A}^{ij}(x+i,y+j)\Big).
\end{aligned} \tag{12}$$

5.3 Analysis and comparison

Direct 2D convolution with a general non-separable $k\times k$ kernel requires $2k^2n^2$ operations. Wavelet convolution with the same kernel takes $2\sum_{i,j=1}^{k}|T(\mathbf{A})^{ij}| \approx 2k^2|T(\mathbf{A})|$ operations. Reconstruction requires an additional $\frac{8}{3}n^2$ operations. This means that wavelet convolution outperforms the general non-separable case even with lossless compression.

In the special case of separable kernels, direct convolution takes only $4kn^2$ operations. Wavelet convolution can also take advantage of a separable kernel. Starting with the 1D wavelet transforms for the shifted rows of the image, we first perform n 1D wavelet convolutions, then reconstruct each row. Then we compute the 1D transforms of the shifted columns, and reconstruct again. The total number of operations in this case is $4k|T(\mathbf{A})| + 2(k+2)n^2$. In order to outperform direct separable convolution, the wavelet decomposition must be sparser by a factor of 6 for $k = 3$, a factor of 3.3 for $k = 5$, and a factor of 2.8 for $k = 7$.

When $k = O(n)$ it is more efficient to perform the convolution in the frequency (Fourier) domain. There are numerous variants of the FFT [16, 26], the most efficient of which is the split-radix FFT algorithm. Using this algorithm, FFT convolution is performed with a total of $6n^2 \log n + 11n^2$ real multiplications. For practical image sizes, FFT convolution is faster than direct convolution only for kernels with $k \geq 7$.

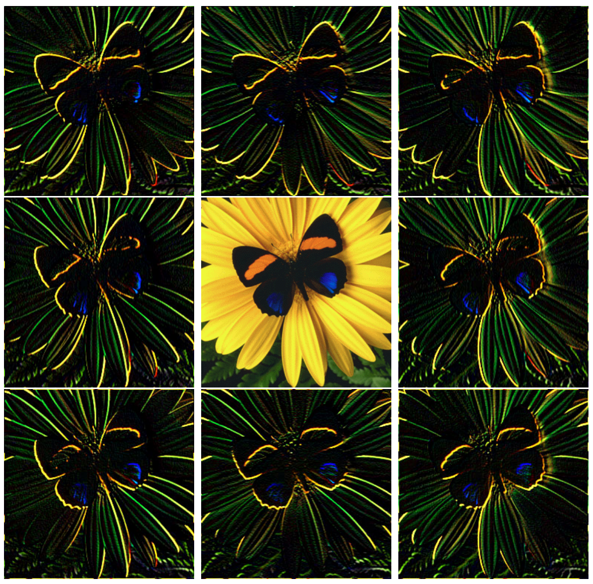


Figure 10: Wavelet convolution with a Sobel filter

Experimental comparison

We experimentally validate our analysis by comparing the performance of direct convolution, wavelet convolution (eq. 11), and wavelet convolution using symmetry (eq. 12). We chose at random 102 images from a stock photography collection of 1015 natural images from various categories. The images were taken at two different resolutions, originally at 640×384 and down sampled to 320×192 with bicubic interpolation. All images were transformed from RGB to the YIQ color space. In order to enable efficient L2 cache utilization convolution was perfomed using tiles of sizes 96×80 and 80×64 pixels. The tiles overlap each other by the convolution filter size to handle boundaries correctly. The images were convolved with Sobel edge-detection filters [18] of three different sizes.

These filters are non-separable and have integer coefficients. The results convolving one example image (identical for the three types of convolution) with eight different filter orientations are shown in Figure 10. The mean convolution times and the corresponding standard deviations for the 102 images are reported in Table 5.

Kernel:	3×3		
Method:	DC	WC	SC
640×384	141.8 (0.2)	69.3 (10.1)	66.4 (5.9)
320×192	35.1 (2.1)	21.5 (3)	19.4 (1.3)

Kernel:	5×5		
Method:	DC	WC	SC
640×384	301.9 (4.5)	121.8 (22.5)	115.1 (9.8)
320×192	75.1 (5)	41.1 (7.9)	33.1 (2.4)

Kernel:	7×7		
Method:	DC	WC	SC
640×384	685.5 (0.5)	337 (65.2)	271 (23.4)
320×192	170.4 (1.4)	114.4 (22.3)	72.8 (6.1)

Table 5: An experimental comparison of direct vs. wavelet convolution. The DC columns report direct convolution times, the WC stands for wavelet-domain convolution, the SC stands for wavelet-domain convolution using symmetry. All times are in milliseconds for a single color channel. The WC and SC times include the final reconstruction step, necessary for displaying the convolved result.

The DC columns report direct convolution times, WC stands for wavelet-domain convolution, and SC stands for wavelet-domain convolution using symmetry. All times are reported in milliseconds for a single color channel. The WC and SC times include the reconstruction step necessary for displaying the convolved result.

The wavelet transform used for this experiment is the S transform. When this transform is applied to the image and its shifts, roughly half of the resulting coefficients are zero. If no further coefficients are zeroed out, the results of DC, WC and SC are identical, but WC and SC can be computed much faster because the computation involves only the non-zero coefficients. In our experiments, the mean speedup of WC over DC is by a factor of 1.9, and the mean speedup of SC over DC is by a factor of 2.3. This includes reconstructing the resulting image by applying the inverse wavelet transform.

Preprocessing and shift invariance

Since our wavelet transform is not shift-invariant, in order to use WC we must first precompute the wavelet decompositions of all k^2 shifts of the image. Decomposition and reconstruction take the same amount of time, so, for example, using WC (eq. 11), it takes 320 milliseconds to precompute 5^2 decompositions of a 640×384 pixel image. This preprocessing time is quite reasonable: for example, in an image processing application, preprocessing can be done when loading the image from the disk or while a filter selection dialogue is being opened, in which case the added time will not be noticeable to the user. Note that once we have transformed all of the k^2 shifts, we can rapidly compute the convolution with *any kernel* of size k or less, as illustrated in Figure 10.

When using SC (eq. 12) the wavelet transform is only applied to the image multiplied by the kernel sum. Since the transfrom is shift covariant it is not applied to the translated differences. The advantage of using SC over WC is that the memory footprint is reduced, only half of the sparse difference images are processed. The disadvantage of SC is limited support for multi-resolution and progressive convolution.

Shift-invariance in wavelet transforms has been investigated by several researchers [19, 9, 33]. For example, the à trous algorithm [20, 17, 33] computes the wavelet transform by using filter banks in which the filter for scale j is the original filter with $2^j - 1$ zeros inserted between every two neighboring samples. When applied *without subsampling* (undecimated) it is shift invariant. Unfortunately, all of the solutions proposed so far are too expensive for our purposes.

3D convolution of an image sequence

Figure 11 shows several frames from a sequence of 128 images of size 256^2 to which we have applied a 3D filter of size $3 \times 3 \times 3$. The filter was designed to enhance temporal edges in the sequence. The original frames are shown in the top row, and the convolved frames in the bottom row. (The entire original and convolved movies can be found under `http://www.cs.huji.ac.il/~danix/wavops`.) In this case, wavelet-domain convolution (after preprocessing) is faster by a factor of 1.8 than direct 3D convolution.

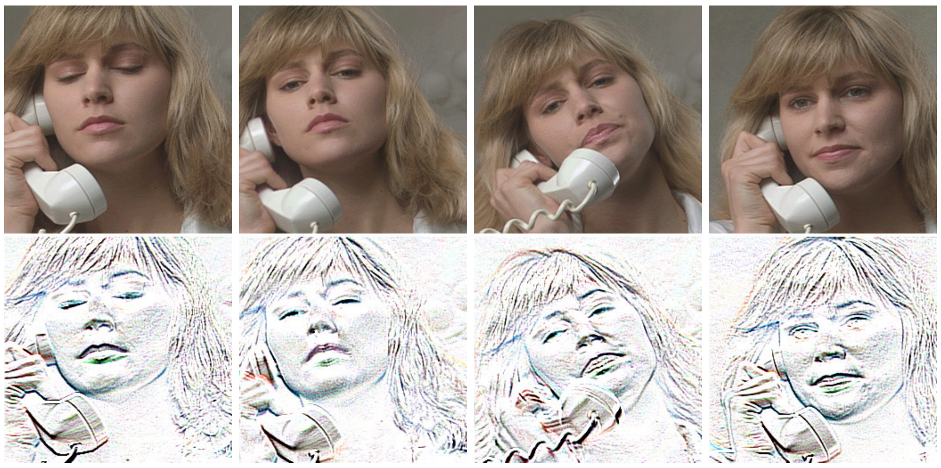


Figure 11: 3D wavelet convolution of an image sequence.

5.4 Visually accurate convolution

In many applications, numerical accuracy is not the primary objective. For example, when previewing the result of a filtering operation, visual accuracy and fast response time is much more important to the user than numerical accuracy. With wavelet convolution it is possible to significantly speed up the operation by zeroing out all coefficients in the wavelet transformed matrices whose magnitude is smaller than some specified threshold. In this way, numerical accuracy is traded off for speed, while the visual quality is gracefully degraded. Figure 12 demonstrates this trade-off by plotting obtained speedup vs. the RMS error between the approximate and the exact convolution results. There are two plots: one using the S transform and another using the TS transform. As far as the RMS error is concerned, the S transform always performs better than the TS transform (because reconstruction is faster). However, a visual comparison indicates that the TS transform is sometimes visually closer to the accurate result.

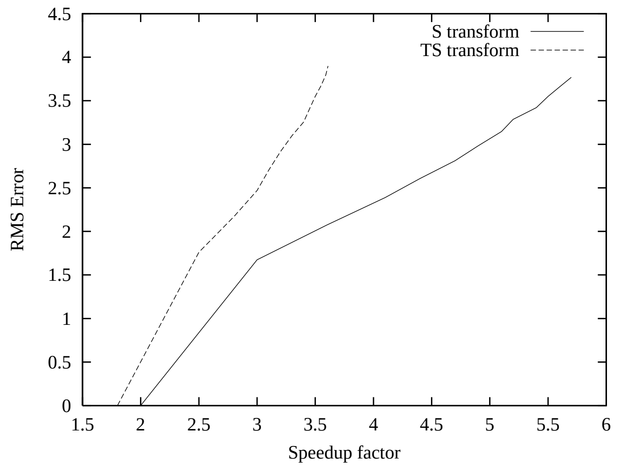


Figure 12: Speedup versus error. The data for this plot was obtained by averaging over 102 randomly chosen images from a database of 1015 stock photographs.

A visual comparison is shown in Figure 13. The exact convolution result is shown in the image on the left (obtained using exact wavelet convolution with a speedup of 1.9 using the S transform and a speedup of 1.5 using the TS transform). The middle column shows the approximate results obtained with S (top) and TS (bottom) transforms. The corresponding speedups are 3 and 2.4, respectively. At this level of compression the approximate results are nearly indistinguishable from the exact result. Further compression yields the images in the right column, with speedups of 6.3 (S) and 4.3 (TS). This time, more errors are visible. Note that the TS image is smoother, so it appears more accurate visually, despite its higher RMS error. Even with these visible errors, note that the main features of the resulting image, i.e., the strong edges, appear correctly in the approximate results. Thus, for some applications, even this level of compression might be quite acceptable.

The threshold below which all wavelet coefficients can be zeroed out without causing noticeable artifacts in the result depends on the type of convolution kernel. For example,

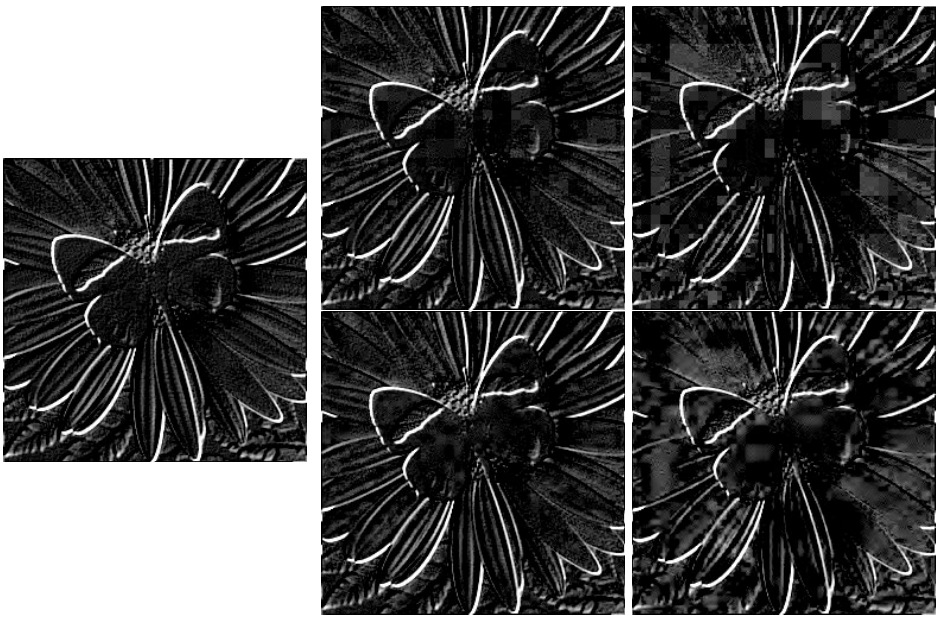


Figure 13: A comparison of visual accuracies.

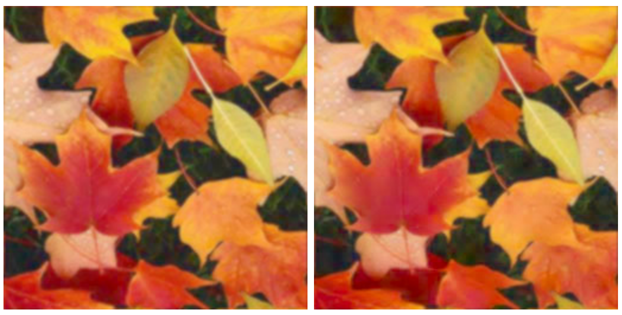


Figure 14: Visually accurate lowpass filtering.

edge detection and sharpen (high pass) filters respond to high frequencies in the image, so zeroing out too many coefficients will eventually cause important edges to disappear, or to appear altered, in the resulting image. On the other hand, smoothing (low pass) filters suppress the frequencies represented by small scale wavelet coefficients, so we can zero coefficients out in a much more liberal fashion, resulting in larger speedups. For example, compare the two images shown in Figure 14. The left image is an exact convolution with a 5×5 Gaussian low pass filter. For this image, wavelet convolution using the TS-transform is 1.5 times faster than direct convolution. The image on the right was computed using the same transform, and 77 percent of the smallest magnitude nonzero coefficients were zeroed out. As a result, the computation is 6.25 times faster than direct convolution, yet there are no visible differences between the two images.

5.5 Local and multi-scale convolution

The space- and frequency-localization properties of the wavelet transform enable us to very easily restrict convolutions to a given spatial window, and perform them on a an arbitrary subset of the different possible scales. Such flexibility is available neither with direct, nor with Fourier convolutions.

Operating on a single scale or on a range of scales is trivial in wavelet domain convolution: simply compute the linear combination in eq. (11) only for wavelet coefficients on the corresponding level(s), i.e., the remaining coefficients are unchanged.

In order to enable spatially restricted wavelet convolution, the decompositions $T(\mathbf{A}^{ij})$ must not be performed to the finest possible level, but rather stopped several levels earlier. The operation is then performed using all of the coefficients inside the window of interest (enlarged by the size of the kernel on each side). Note that performing spatially restricted wavelet convolution in this manner is only correct when using in-place wavelet transforms.

Figure 15 illustrates local and multi-scale convolution. An image has been wavelet-convolved with a Laplacian-Prewitt filter in several different manners. In the top left image the convolution is restricted to a spatial window and at the same time performed only on coefficients of a single scale. In the top right and bottom row of the figure, we show wavelet convolution with the *same* kernel applied to an increasing range of scales. In the top right image the kernel is applied to all wavelet coefficients except those of the three finest scales, in the bottom left image the kernel is applied to all but the two finest scales, and in the bottom right image the kernel is applied to all but the finest scale. The results of the convolution were all thresholded using the same value. By restricting the operation of the same filter to different subsets of the coefficients of the same transform we can detect and isolate features at different scales.

6 Conclusions and Future Work

We have presented a simple way of computing operations such as image blending, 3D image warping, and image convolution in the wavelet domain. We have demonstrated (both analytically and experimentally) that performing these operations in the wavelet domain is in many cases faster than their direct computation. Furthermore, wavelet domain operations enable progressive and multi-resolution computations, as well as space and frequency locality. We have demonstrated our approach both on still images and on image sequences.

In order to extend and improve our approach, we would like to develop an adaptive multiresolution scheme, which would allow to operate upon different regions of an image at different resolutions.

In the future we plan to explore the new applications of our approach to other types of image and video operations, such as other types of image warping (perhaps using more complicated mappings).

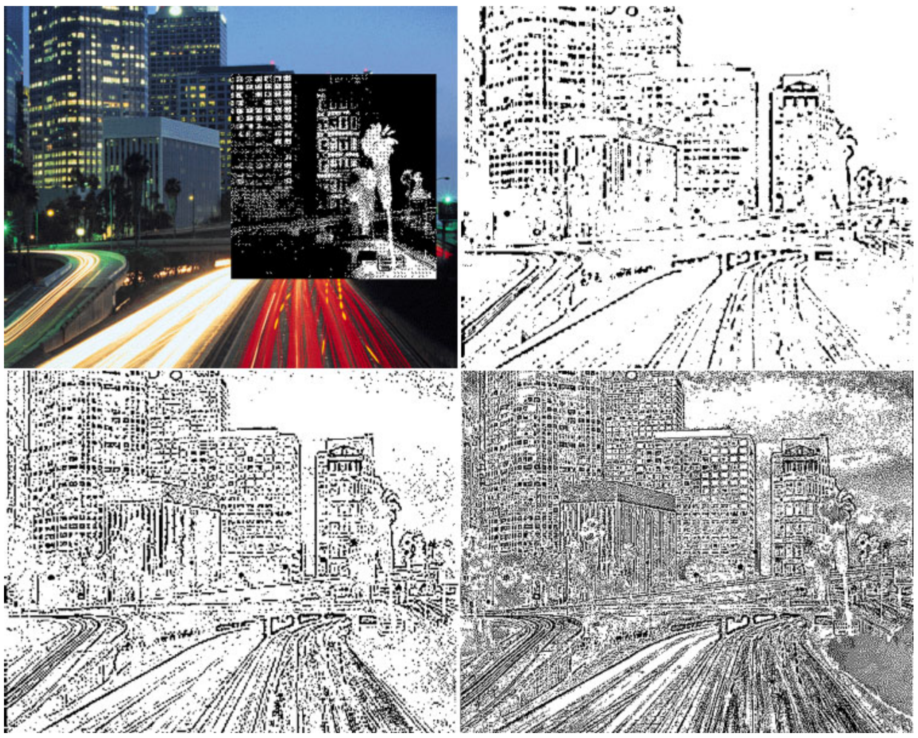


Figure 15: Localized (top left) and multi-scale wavelet convolution.

References

[1] Ronen Basri and David Jacobs. Lambertian reflectance and linear subspaces, 2001.

[2] G. Beylkin. On the representation of operators in bases of compactly supported wavelets. *SIAM Journal of Numerical Analysis*, 29:1716–1740, 1992.

[3] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure Appl. Math.*, 44:141–183, 1991.

[4] Peter J. Burt and Edward H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[5] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. *Appl. Comput. Harmon. Anal.*, 5(3):332–369, 1998.

[6] Shenchang Eric Chen. QuickTime VR — an image-based approach to virtual environment navigation. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '95)*, pages 29–38, 1995.

[7] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '93)*, pages 279–288, 1993.

[8] A. Cohen, I. Daubechies, and J. Feauveau. Bi-orthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Math*, 45:485–560, 1992.

[9] I. Cohen, S. Raz, and D. Malah. Orthonormal shift-invariant wavelet packet decomposition and representation. *Signal Processing*, 57(3):251–270, March 1997.

[10] W. J. Dally, L. McMillan, G. Bishop, and H. Fuchs. The delta tree: An object-centered approach to image-based rendering. MIT AI Lab Technical Memo 1604, MIT, May 1996.

[11] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.

[12] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 145–156, Los Angeles, 2000. Addison Wesley Longman.

[13] Ronald A. DeVore, Björn Jawerth, and Bradley J. Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2 (Part II)):719–746, March 1992.

[14] Andrew Dorrell and David Lowe. Fast image operation in orthogonal and overcomplete wavelet spaces. In *3rd Conference on Digital Image Computing Techniques and Applications*, Brisbane, Australia, December 1995.

[15] Iddo Drori and Dani Lischinski. Wavelet warping. In Bernard Péroche and Holly Rushmeier, editors, *Rendering Techniques 2000*, pages 113–124. Springer-Verlag, June 2000.

[16] P. Duhamel and M. Vetterli. Fast Fourier transforms: A tutorial review and a state of the art. *Signal Proc.*, 19(4):259–299, April 1990.

[17] P. Dutilleux. An implementation of the "algorithme à trous" to compute the wavelet transform. In Jean-Michel Combes, Alexander Grossman, and Philippe Tchamitchian, editors, *Wavelets: Time-Frequency Methods and Phase Space*, Inverse Problems and Theoretical Imaging, pages 298–304, Berlin, 1989. Springer-Verlag.

[18] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[19] H. Guo and C.S. Burrus. Convolution using the undecimated discrete wavelet transform. In *Proc. Int. Conf. Acoust., Speech, Signal Processing (ICASSP-96)*, Atlanta, GA, May 7–10 1996.

[20] M. Holschneider, R. Kronland-Martinet, J. Morlet, and Ph. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In Jean-Michel Combes, Alexander Grossman, and Philippe Tchamitchian, editors, *Wavelets: Time-Frequency Methods and Phase Space*, Inverse Problems and Theoretical Imaging, pages 286–297. Springer-Verlag, Berlin, 1989.

[21] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.

[22] William R. Mark, Leonard McMillan, and Gary Bishop. Post-rendering 3D warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, April 1997.

[23] Leonard McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, 1997.

[24] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '95)*, pages 39–46, 1995.

[25] Jeffry S. Nimeroff, Eero Simoncelli, and Julie Dorsey. Efficient re-rendering of naturally illuminated environments. In *Fifth Eurographics Workshop on Rendering*, pages 359–373, Darmstadt, Germany, 1994.

[26] H. J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer-Verlag, Berlin, 1982.

[27] Joan M. Ogden, Edward H. Adelson, J.R. Bergen, and Peter J. Burt. Pyramid-based computer graphics. *RCA Engineer*, 30(5):4–15, 1985.

[28] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Siggraph 2001, Computer Graphics Proceedings*, Los Angeles, 2001.

[29] Ramana Rao. Nonstandard wavelet decomposition of the convolution and the derivative operators. *CNLS Newsletter*, September 1997. http://cnls.lanl.gov/Highlights/1997-09/.

[30] Julien Reichel. *Complexity Related Aspects of Image Compression*. PhD thesis, Swiss Federal Institute of Technology, February 2001.

[31] A. Said and W. A. Pearlman. An image multiresolution representation for lossless and lossy image compression. *IEEE. Trans. Image Processing*, 5(9):1303–1310, 1996.

[32] Jonathan W. Shade, Steven J. Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In Michael Cohen, editor, *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '98)*, pages 231–242, July 1998.

[33] Mark J. Shensa. The discrete wavelet transform: Wedding the à trous and Mallat algorithms. *IEEE Transactions on Signal Processing*, 40(10):2464–2482, 1992.

[34] Brian C. Smith and Lawrence A. Rowe. Algorithms for manipulating compressed images. *IEEE Computer Graphics and Applications*, 13(5):34–42, September 1993.

[35] Brian C. Smith and Lawrence A. Rowe. Compressed domain processing of JPEG-encoded images. *Real-Time Imaging*, 2:3–17, 1996.

[36] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1996.

[37] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal of Mathematical Analysis*, 29(2):511–546, 1997.

[38] A. Zandi, M. Boliek, E. L. Schwartz, and M. J. Gormish. Crew lossless/lossy medical image compression. Technical Report CRC-TR-9526, RICOH California Research Center, 1995.

[39] 3DV Systems. http://www.3dvsystems.com.