Wavelet Warping

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

Abstract. We present *wavelet warping* — a new class of forward 3D warping algorithms for image-based rendering. In wavelet warping most of the warping operation is performed in the wavelet domain, by operating on the coefficients of the wavelet transforms of the images and other matrices defined by the mapping. Operating in this fashion is often more efficient than performing the 3D warp in the standard manner. Perhaps more importantly, operating in the wavelet domain allows one to perform the 3D warping operation progressively and to generate target views at multiple resolutions. We describe wavelet warping of planar, cylindrical, and spherical reference images and demonstrate that the resulting algorithms compare favorably to their standard counterparts. We also discuss and demonstrate utilization of temporal coherence when wavelet-warping image sequences.

1 Introduction

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.

This paper presents *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 combination on the sparse wavelet transform coefficients. Applying the inverse wavelet transform to the resulting coefficients yields the desired linear combinations. Operating in the wavelet domain is advantageous in several aspects:

Sparse representation: The wavelet decomposition of an image or a large matrix is typically much sparser (has fewer non-zero coefficients) than the original direct representation. This property has been utilized to speed up various numerical operations [1, 2]. In our case, a sparse representation of the matrices results in faster computation of their linear combinations since we operate only on the non-zero coefficients.

Multi-resolution and progressive computation: The wavelet decomposition represents each matrix at multiple scales. Such a representation makes it easy to perform the warping operation at multiple resolutions, as well as in a progressive, coarse-to-fine fashion.

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.

1.1 Contributions

We describe in detail wavelet warping algorithms for three common types of 3D image warps: planar-to-planar warp, cylindrical-to-planar warp, and spherical-to-planar warp. 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, e.g., [4, 9]. By using depth information, a 3D warper enables users to change the *viewing position* (center of projection), in addition to the viewing direction [13]. 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 multi-resolution and progressive 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 quadtree), but at a cost of increasing the size of the representation. In addition, wavelet warping has the advantage that the computation is progressive: a low resolution result can be progressively refined without redundant computations.

Our final contribution is a new algorithm for 3D 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 Related work

The idea of representing a scene as a set of reference images was introduced to computer graphics by Chen and Williams [5] and by McMillan and Bishop [13]. The equations of 3D warping are developed in detail in McMillan's PhD thesis [12]. Mark *et al.* [11] and Shade *et al.* [14] discuss different frameworks for image-based rendering and warping. Dally *et al.* [6] 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 [15] 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 [16] 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 faster and more flexible 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.

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 [10]. Wavelets have also been applied to a wide variety of problems in computer graphics [17]. In this section we briefly review wavelet-related terminology and concepts that will be used later in the paper.

Lifting: The lifting scheme [18] 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 [7], thus allowing: (i) in-place computation of the wavelet transform; (ii) faster computation, asymptotically reducing the complexity by a factor of two; (iii) construction of wavelet transforms that map integers to integers [3].

Integer wavelets: Integer wavelet transforms operate on integer valued signals to produce integer valued wavelet coefficients. Such transforms have been effectively used for lossless compression of images [3]. Calderbank *et al.* [3] describe invertible integer wavelet transforms, but use floating point arithmetic to compute them. In our implementation, the integer transforms are computed using integer arithmetic, with addition, subtraction and shift operations only.

2D transforms: 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*. The nonstandard decomposition is computed by applying the 1D transform alternating between rows and columns of the matrix. In this paper we use both the 1D wavelet transform and the 2D non-standard wavelet transform.

Linear combinations of matrices

Our approach is based on fast 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

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

In other words, $\mathbf{A}$ can be computed in the wavelet domain, by precomputing 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 $T(\mathbf{A}_i)$ are sparse, this computation can be done rapidly by operating only on the non-zero coefficients of each transform.

3 3D Warping

This section describes the application of our approach to the 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 [12] 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 [12]:

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

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

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

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 a spherical reference image to a planar target image is defined by:

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

3.1 Wavelet warping

In order to perform 3D warping in the wavelet domain, we express the warping equations as element-wise divisions between three linear combinations of 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

$$\mathbf{U} = \frac{\mathbf{F}_1}{\mathbf{F}_3} \quad \text{and} \quad \mathbf{V} = \frac{\mathbf{F}_2}{\mathbf{F}_3},$$

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 simply 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 $\mathbf{A}$ 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 $\mathbf{B}$ is a linear ramp, and all of its columns are the same vector. The matrix $\mathbf{C}$ 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 $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ 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 $\mathbf{A}$ and $\mathbf{B}$ is a function of a single variable x, which means that in each of these two matrices all of the rows are equal. Similarly, $\mathbf{C}$ 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 $\mathbf{C}$ is a function of a single variable y, and therefore all of its columns are equal.

The wavelet warping operation consists of two steps: computation of linear combinations of matrices (equation (6)), followed by clipping and element-wise divide. The first step is carried out in the wavelet domain, as illustrated in Figure 1. 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 $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ 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.

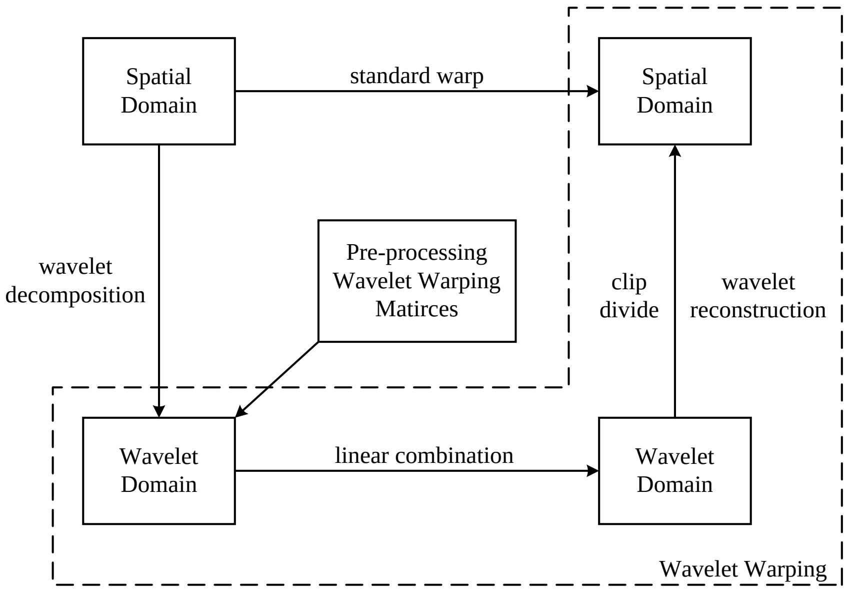

Fig. 1. Standard warping vs. wavelet warping.

- 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 view, and are calculated once for each target view, the same as in a standard warp.

The pseudocode for the resulting warping algorithm is listed in Figure 2. Note that all wavelet-transformed matrices are represented by arrays containing only the non-zero coefficients. Each entry in such an array has an *index* field indicating position of the coefficient, and a *value* field containing the value of the coefficient. Our implementation uses an integer wavelet transform [3]. Since 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 integer range.

3.2 Choice of basis

In order to make our wavelet warping algorithm as efficient as possible, we must choose a suitable wavelet basis. There are two requirements that the chosen basis must satisfy: (i) the transforms $T(\mathbf{A})$, $T(\mathbf{B})$, $T(\mathbf{C})$, and $T(\mathbf{D})$ should be sparse; (ii) the reconstructions $\mathbf{F}_i = T^{-1}(\hat{\mathbf{F}}_i)$ should be fast to compute. After experimenting with several different options, we have chosen a slightly modified version of the second order interpolating wavelet transform, $I(2,2)$ [18]. The modification consists of omitting the update phase of the lifting scheme. The resulting transform requires $\frac{8}{3}n^2$ operations to decompose or reconstruct an $n \times n$ matrix using the 2D nonstandard wavelet transform.

The wavelet coefficients of this transform measure the extent to which the original function fails to be linear. In the case of a planar warp, the matrices **A** and **B** are simply

```
Pre-processing:
    For each type of warp, precompute transforms: $T(\mathbf{A}), T(\mathbf{B}), T(\mathbf{C})$
    For each reference image, precompute $T(\mathbf{D})$
    Store non-zero coefficients in arrays $t_A, t_B, t_C, t_D$

Input:   Reference image, target view, coefficient arrays $t_A, t_B, t_C, t_D$
Output:  Target image

3D Warp:
    for $i = 1$ to 3
        Compute $m_{ij}$ for $j = 1$ to 4
        foreach coefficient array $t$
            for $l = 1$ to length of $t$
                $\hat{\mathbf{F}}_i[t[l].index] = \hat{\mathbf{F}}_i[t[l].index] + m_{ij}t[l].value$
            endfor
        endfor
        Reconstruct $\mathbf{F}_i = T^{-1}(\hat{\mathbf{F}}_i)$
    endfor

Clip and Divide:
    for $x = 0$ to $w - 1$
        for $y = 0$ to $h - 1$
            $w = \mathbf{F}_3(x, y)$
            if $w > 0$ then $(u, v) = \frac{1}{w}(\mathbf{F}_1(x, y), \mathbf{F}_2(x, y))$
            if $(u, v) \in$ target image space then $target(u, v) = ref(x, y)$
        endfor
    endfor
```

Fig. 2. Wavelet domain warping.

linear ramps and matrix **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 $\frac{1}{9}n^2$ non-zero 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 **D**, the number of non-zero coefficients depends, of course, on the reference image. In our experiments, roughly one third of the coefficients of $T(\mathbf{D})$ 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 Analysis and Results

The complexity of the standard 3D warp depends on the type of warp. 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 [14]. 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, the computation cannot be done incrementally. To perform the standard warp efficiently, the terms in matrices **A**, **B**, and **C** are precomputed and stored in lookup tables (LUT). In the cylindrical-to-planar case, further savings are possible since, 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 multiplications and $t - |T(\mathbf{D})|$ additions (where t is the total number of non-zero wavelet 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 the following table:

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

As explained in Section 3.2, our choice of wavelet basis results in a total number of $t = |T(\mathbf{D})| + 5$ non-zero 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 a factor greater than $20/13$.

4.1 2D vs. 1D decomposition

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 may exceed the size of the L2 cache, resulting in a performance penalty. In this case, we use a modified version of the wavelet warping algorithm. This version computes a 1D wavelet decomposition of each row in each of the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and $\mathbf{D}$ instead of the 2D non-standard wavelet decomposition. The linear combination of the matrices is then computed row-by-row. Now the reconstruction step is performed on one row at a time, and the required intermediate storage is $O(n)$. In addition to avoiding cache misses, this modification also allows us to take advantage of the fact that in many of the matrices the rows are identical, as pointed out earlier for the standard cylindrical and spherical warps. On the other hand, using the 1D transform version slightly increases the number of non-zero coefficients: in our experiments with this method we found that roughly half of the coefficients of $T(\mathbf{D})$ were non-zero (compared to third with 2D decomposition). The 1D version is still faster than the 2D version, but it compromises our ability to perform the warping in a true multi-resolution fashion, as described in the next section. Since memory speed and cache size tend to increase faster than image resolution, it is safe to assume that in the near future it will be possible to use the 2D version even on large images, without incurring a performance penalty.

4.2 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, 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 the following table.

Type of Warp	standard warp	2D wavelet	1D wavelet
planar (512^2 reference)	6.5	6.5	7
cylindrical (512×256 reference)	12	12	15
spherical (512×256 reference)	7.7	14	14
spherical (1024×512 reference)	4	4	6.5

As predicted by our analysis, in the planar and cylindrical cases, we found wavelet warping to be roughly as fast as the standard algorithms, when 2D decomposition of the matrices was used. Using the 1D decomposition version, we found wavelet warping to be slightly faster (up to 25 percent in the cylindrical case) than the standard algorithms. 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 with a 512×256 reference image, wavelet warping outperforms the standard algorithm by a factor of roughly 1.8. When the resolution of the spherical reference image is increased to 1024×512, 2D wavelet warping suffers from cache misses and the performance drops down to the speed of the standard algorithm. The 1D wavelet version, however, still outperforms the standard algorithm by a factor of over 1.6. Figure 3 (see color plates) shows a spherical-to-planar warping example. The two rectangular images on the left show a spherical reference image of a chapel along with the corresponding depth image. The middle image is a planar view of the chapel taken from a slightly displaced view point. Since only one reference image was used, some disocclusion artifacts can be seen. The right image is another planar view of the chapel, taken from the original view point.

5 Multi-Resolution Warping

Suppose that we have an image-based representation of a 3D object and would like to generate a novel view in which the 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 when the view is constantly 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, we compute the linear combination (10) exactly as before. However, in this case there is no need to perform a full reconstruction of the result. For example, if the target image resolution is twice smaller in each dimension, we stop reconstruction just before processing the wavelet coefficients of the finest level. As a result, the reconstruction step 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.

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 4 (see color plates) 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. Both reference views and their corresponding depth images are 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 is performed at roughly 6.5 frames per second (see table in previous section). However, when the target resolution is reduced by half, wavelet warping becomes more than 4 times faster compared to full resolution warping (around 27 frames per second). At quarter resolution, wavelet warping becomes more than 16 times faster.

It should be noted that multi-resolution warping can also be achieved within a standard warping framework by using an over-complete pyramid-based image representation (e.g. a quadtree), but at a cost of increasing the size of the representation by a factor of $4/3$. In addition, wavelet warping has the advantage that the computation is progressive: a low resolution result can be progressively refined without redundant computations, simply by performing one more level of reconstruction. In contrast, in a pyramid-based scheme each level must be warped from scratch.

6 Fast Warping of Image Sequences

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:

$$\mathbf{F}_i^{(t+1)} = \mathbf{F}_i^{(t)} + T^{-1}\left(m_{i4}T(\Delta\mathbf{D}^{(t)})\right). \tag{11}$$

In other words, the differences between successive disparity images are multiplied by m_{i4} in the wavelet domain. Since the disparities of many pixels remain 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 target image sequence by warping sequences of images from three different sources into a common target view. Three source images, one from each source, are shown in the top row of Figure 5 (see color plates). Three of the resulting images are shown in the bottom row. One of the sources is a video sequence of an actor performing in front of a Zcam — a real-time depth-sensing camera [19]. Another source is a synthetic animation of a coffee-table following a circular trajectory. The third source is a still image of a synthetic 3D scene (a room). 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. 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 wavelet-warping each frame individually.

7 Conclusions and Future Work

We have presented a simple way of computing various 3D image warps in the wavelet domain. We have demonstrated (both analytically and experimentally) that performing these warps in the wavelet domain is in many cases faster than their direct computation, particularly in the spherical-to-planar warp case. Furthermore, wavelet warping enables multi-resolution and progressive computations, with no storage or computation overhead. Finally, we have presented a wavelet warping algorithm for image sequences, which utilizes temporal coherence to achieve considerable speedups over frame-by-frame warping. We intend to use this algorithm for 3D warping of spherical depth movies.

In order to extend and improve our wavelet warping approach, we would like to develop an adaptive multi-resolution warping scheme, which would allow to warp different regions of a reference view at different resolutions.

The warping algorithms presented in this paper are one example of a more general approach in which various operations on images are expressed using linear combinations of matrices, and then performed directly in the wavelet domain. In addition to 3D warping we have applied this approach to convolution of images and image sequences [8]. In the future we plan to apply our approach to other types of image and video operations, such as other types of image warping (perhaps using more complicated mappings), and blending of image sequences.

Acknowledgements

This research was supported by the Israel Science Foundation, founded by the Israel Academy of Sciences and Humanities. We would also like to thank 3DV Systems [19] and Eyal Ofek for providing us with the depth video used in Section 6. Finally, thanks to the Siggraph and EGWR anonymous reviewers, whose comments have greatly helped to improve this paper.

References

1. G. Beylkin. On the representation of operators in bases of compactly supported wavelets. *SIAM Journal of Numerical Analysis*, 29:1716–1740, 1992.
2. G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure Appl. Math.*, 44:141–183, 1991.
3. 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.
4. S. E. 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.
5. S. E. Chen and L. Williams. View interpolation for image synthesis. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '93)*, pages 279–288, 1993.
6. 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.
7. I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.
8. I. Drori. Image operations in the wavelet domain. Master's thesis, School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel, Jan. 2000.
9. Internet Pictures Corporation (iPIX). http://www.ipix.com.
10. S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.
11. W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3D warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, Apr. 1997.
12. L. 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.
13. L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '95)*, pages 39–46, 1995.
14. J. W. Shade, S. J. Gortler, L. He, and R. Szeliski. Layered depth images. In M. Cohen, editor, *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '98)*, pages 231–242, July 1998.
15. B. C. Smith and L. A. Rowe. Algorithms for manipulating compressed images. *IEEE Computer Graphics and Applications*, 13(5):34–42, Sept. 1993.
16. B. C. Smith and L. A. Rowe. Compressed domain processing of JPEG-encoded images. *Real-Time Imaging*, 2:3–17, 1996.
17. E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1996.
18. W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal of Mathematical Analysis*, 29(2):511–546, 1997.
19. 3DV Systems. http://www.3dvsystems.com.