The Multi-Dimensional Hartley Transform as a Basis for Volume Rendering

Thomas Theußl, Robert F. Tobler and Eduard Gröller
Institute of Computer Graphics
Vienna University of Technology
Karlsplatz 13/186, A-1040 Wien, Austria
http://www.cg.tuwien.ac.at/home/
e-mail: {theussl,rft,groeller}@cg.tuwien.ac.at

ABSTRACT

The Fast Hartley Transform (FHT), a discrete version of the Hartley Transform (HT), has been studied in various papers and shown to be faster and more convenient to implement and handle than the corresponding Fast Fourier Transform (FFT). As the HT is not as nicely separable as the FT, a multidimensional version of the HT needs to perform a final correction step to convert the result of separate HTs for each dimension into the final multi-dimensional transform. Although there exist algorithms for two and three dimensions, no generalization to arbitrary dimensions can be found in the literature. We demonstrate an easily comprehensible and efficient implementation of the fast HT and its multi-dimensional extension. By adapting this algorithm to volume rendering by the projection-slice theorem and by the use for filter analysis in frequency domain we further demonstrate the importance of the HT in this application area.

Keywords: Hartley transform, Fourier transform, volume rendering.

1 Introduction

We usually look at signals as a function of values at certain times, which are then said to be in spatial domain. However, it is sometimes more useful to look at them as a function of magnitude and phase at certain frequencies. We call signals specified that way to be in frequency domain. This way of looking at functions allows us, for example, to explain quite easily a fundamental concept in computer graphics: sampling and reconstruction and inevitable aliasing (or reconstruction error) [2, 3]. Investigation of functions in frequency domain is also of great importance in, e.g., digital signal processing, electronic engineering or mathematics.

There exist a lot of approaches that transform data to the frequency domain. The most widely known and common used one is the FT. This technique was invented by Jean-Baptiste-Joseph Fourier who used it to describe conduction of heat in solid bodies. Another way to change between spatial and frequency domain is the use of wavelets, which are also quite popular in computer graphics [22] and have been used for, e.g., image and surface compression and editing or multiresolution surface modeling.

We will use a technique which is quite similar to Fourier's, but generates only real instead of complex output. Since most data is real in computer graphics (e.g., images) and especially volume rendering (volumetric data sets) it is easier and more intuitive to use. It is named after Brian V. L. Hartley, who conceived it in 1942 [11]. We will demonstrate its usefulness in the area of volume rendering.

2 The Hartley Transform

In this section we will give definition and properties of the various types of HTs, specifically the general HT in Section 2.1, the discrete HT in Section 2.2 and the FHT in Section 2.3. In Section 3 we will introduce our generalization of the HT to arbitrary dimensions. To demonstrate to usefulness of the HT in volume rendering we will present two example applications in Section 4 and finally we will conclude in Section 5.

2.1 The General Hartley Transform

The HT takes a real-valued function $h(t)$ through a similar integral than the FT:

$$H(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} h(t)(\cos\omega t + \sin\omega t)dt \qquad (1)$$

The limitations on the existence of this transform are similar to the Fourier transform, however the HT of a real-valued function is also real-valued. In addition to that Hartley showed that the HT is its own inverse:

$$h(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} H(\omega)(\cos\omega t + \sin\omega t)d\omega \qquad (2)$$

For a long time the HT was seen as interesting, but of no practical value until it was shown that it could be made to serve all the purposes that were previously the exclusive domain of the FT [4].

2.2 The Discrete Hartley Transform

The practical application of the HT is based on the discrete HT which takes a vector of n real values $h(t)$, with $t \in [0, n-1]$ and transforms them into another vector $H(f)$ of n real values, with $f \in [0, n-1]$:

$$H(f) = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} h(t)(\cos\frac{2\pi ft}{n} + \sin\frac{2\pi ft}{n}) \qquad (3)$$

To interpret this transformed vector, you can take $H(f)$ and $H(-f) = H(n-f)$ and think of them as a pair in the Gaussian plane $(H(f) + iH(-f))$. This pair is then a rotated copy of the FT of $h(t)$ with the rotation being $\frac{\pi}{4}$.

2.3 The Fast Hartley Transform

Bracewell [6] and others have shown that the fast version of the discrete HT outperforms the FT with the additional benefit of being only real-valued and thus easier to implement and maintain. We use several optimizations to make our algorithm efficient but nevertheless maintain comprehensibility.

The basis of the FHT is the following recurrence:

$$\begin{aligned} H(f) &= H_{even}(f) \\ &\quad + \cos\frac{2\pi f}{n} H_{odd}(f) \\ &\quad + \sin\frac{2\pi f}{n} H_{odd}(n-f) \end{aligned} \qquad (4)$$

where $H_{even}(f)$ is the HT of a vector of all values with even indices, and $H_{odd}(f)$ is the HT of a vector of all values with odd indices. The termination of the recurrence is supplied by the fact that a vector with only one value is its own HT, which of course only works when the data size is a power of two. If it is not, zeros should be padded up to the next power of two [19].

Since the algorithm is based on performing HTs on the parts of the vector with even indices and odd indices respectively, the standard practice for both FTs and HTs, is to split the algorithm in two parts, a part that rearranges the values in the vector according to their indices being even or odd, and a part that performs the weighted sums given in Eq. 4.

There are two general strategies, based on the order of these two parts of the algorithm [19]. We chose to implement the version which performs the rearrangement of the values first: The necessary rearrangement based on the recurrence is to place each element at the position indicated by the bit-reversal of its index [19].

After this first part the vector contains n Hartley transformed vectors of length 1. Now the part of the algorithm that performs the weighted sums takes each two successive small "transforms" and combines them to one transform of double length. In $log(n)$ passes over the vector these n small transforms of length 1 are thus combined to the complete transform of the original vector.

The weighted combination of two successive small transforms to obtain a single double length transforms makes use of the following facts:

- Due to the cyclic nature of the transform (in order to avoid undesired frequencies the data is assumed to be periodic), the following equivalences hold for all $f \in [0, n-1]$:

$$\begin{aligned} H_{even}(n/2 + f) &= H_{even}(f) \\ H_{odd}(n/2 + f) &= H_{odd}(f) \end{aligned} \qquad (5)$$

- If the weighted combination of the values at $H(f)$, $H(n/2 - f)$, $H(n/2 + f)$, and $H(n-f)$ are performed at the same time, the cosine and sine terms of Eq. (4) need to be evaluated only once for all these four values.

- The combination of these four values can be performed in-place. This is the so-called *butterfly* scheme [12] which is also used in the fast Fourier transform.

The remaining problem to be solved is the fast evaluation of the cosine and sine terms for a number of successive quadruples of values. Press et al. [19] suggest to use the following trigonometric recurrence for this evaluation:

$$\begin{aligned} a &= 2\sin((d/2)^2) \\ b &= \sin d \\ \cos(t+d) &= \cos t - (a \cdot \cos t + b \cdot \sin t) \\ \sin(t+d) &= \sin t - (a \cdot \sin t - b \cdot \cos t) \end{aligned} \qquad (6)$$

Alternatively the cosine and sine values could be tabulated for the needed frequencies. As the standard algorithm only works on arrays of a length being a power of two, only one big table for the maximum possible array length is needed.

The resulting algorithm can be written in a very clear manner and although highly optimized and efficient it stays comprehensible, as demonstrated in Fig. 2.3.

```
void vector_hartley_transform_simple(
    double * v, long size)                    /* size must be a power of 2 */
{
    long old_len = 0;                        /* initialize to avoid warning */
    long len;
    long new_len = 0;                        /* initialize to avoid warning */

    vector_bitreverse_indices(v,size); /* v[b0 b1 .. bn] <-> v[bn .. b1 b0] */

    for (len = 1; len < size; old_len = len, len = new_len)
    {
        long i, j;
        double hi, hj;
        new_len = 2 * len;              /* build transform of double length */

        for (i = 0; i < size; i += new_len)             /* for all blocks */
        {
            j = i + len;                                /* special case: */
            hi = v[i]; hj = v[j];
            v[i] = hi + hj;                             /* f = 0 */
            v[j] = hi - hj;                             /* f = PI */
        }
        if (len < 2) continue;
        for (i = old_len; i < size; i += new_len)       /* for all blocks */
        {
            j = i + len;                                /* special case: */
            hi = v[i]; hj = v[j];
            v[i] = hi + hj;                             /* f = PI/2 */
            v[j] = hi - hj;                             /* f = 3 * PI/2 */
        }
        if (len >= 4)
        {
            double d = MATH_2_MUL_PI / new_len;
            double a = 2.0 * M_SQR(sin(d * 0.5));       /* initialize trig. */
            double b = sin(d);                          /* recurrence */
            double cos_t = 1.0;
            double sin_t = 0.0;
            long f;
            for (f = 1; f < old_len; f++)               /* for all freqs in */
            {                                           /*   the first quad */
                double one = a * cos_t + b * sin_t;     /* trig. recurrence */
                double two = a * sin_t - b * cos_t;
                cos_t -= one;                           /*   cos (t + d) */
                sin_t -= two;                           /*   sin (t + d) */
                for (i = f, j = len - f;                /* for all blocks */
                     i < size;
                     i += new_len, j += new_len)
                {
                    long k = i + len;
                    long l = j + len;
                    one = cos_t * v[k] + sin_t * v[l];
                    two = cos_t * v[l] - sin_t * v[k];
                    hi = v[i]; hj = v[j];
                    v[i] = hi + one; v[k] = hi - one;   /* all four quads */
                    v[j] = hj - two; v[l] = hj + two;   /* (i,j,k,l) */
                }
            }
        }
    }
}
```

Figure 1: A C-implementation of the Fast Hartley Transform

3 The Multi-Dimensional Hartley Transform

The d-dimensional discrete HT of a function $h(t_1, t_2, ..., t_d)$ is defined similarly to the equivalent multi-dimensional FT:

$$H(f_1, f_2, ..., f_d) = \sum_{t_1=0}^{n_1-1} \sum_{t_2=0}^{n_2-1} ... \sum_{t_d=0}^{n_d-1} h(t_1, t_2, ..., t_d) \cdot \cdot \text{ cas } [2\pi(\frac{f_1 t_1}{n_1} + \frac{f_2 t_2}{n_2} + ... + \frac{f_d t_d}{n_d})] \quad (7)$$

Where $\text{cas } \omega = \cos\omega + \sin\omega$.

However the kernel $\text{cas } [2\pi(\frac{f_1 t_1}{n_1} + \frac{f_2 t_2}{n_2} + ... + \frac{f_d t_d}{n_d})]$ is not as easily separable as the corresponding kernel of the FT:

$$\exp[i2\pi(\frac{f_1 t_1}{n_1} + \frac{f_2 t_2}{n_2} + ... + \frac{f_d t_d}{n_d})] = \exp(i2\pi\frac{f_1 t_1}{n_1}) \cdot \exp(i2\pi\frac{f_2 t_2}{n_2}) \cdot ... \cdot \exp(i2\pi\frac{f_d t_d}{n_d}) \quad (8)$$

This separability is the basis of performing the multi-dimensional Fourier transform by successive unidimensional FTs in each dimension.

Bracewell et al. [7] have described a correction scheme to overcome the missing separability of the HT. It is based on the following trigonometric identity:

$$\begin{aligned} 2 \text{ cas } (\alpha + \beta) = & \text{ cas } \alpha \text{ cas } \beta + \\ + & \text{ cas } \alpha \text{ cas } (-\beta) + \\ + & \text{ cas } (-\alpha) \text{ cas } \beta - \\ - & \text{ cas } (-\alpha) \text{ cas } (-\beta) \end{aligned} \quad (9)$$

Using this scheme, the d-dimensional input data is first transformed with the HT, as if it were separable, resulting in a temporary transformation T:

$$T(f_1, f_2, ..., f_d) = \sum_{t_1=0}^{n_1-1} \sum_{t_2=0}^{n_2-1} ... \sum_{t_d=0}^{n_d-1} h(t_1, t_2, ..., t_d) \text{ cas } \frac{2\pi f_1 t_1}{n_1} \cdot \cdot \text{ cas } \frac{2\pi f_2 t_2}{n_2} \cdot ... \cdot \text{ cas } \frac{2\pi f_d t_d}{n_d} \quad (10)$$

Using the casine identity (Eq. 9) $d-1$ passes over the temporary transform (Eq. 10) can be used to successively combine pairs of casine-terms to a single sum of all casine terms. This combination of the terms can be performed in place using a butterfly scheme that performs four combinations in one step.

Nevertheless, this scheme can be improved as Hong Hao and Bracewell [10] have optimized the corresponding step to Eq. 9 for three dimensions, noting that a scheme combining three casine-terms at once uses only four of the eight available terms:

$$\begin{aligned} 2 \text{ cas } (\alpha + \beta + \gamma) = & \text{ cas } (-\alpha) \text{ cas } \beta \text{ cas } \gamma \\ + & \text{ cas } \alpha \text{ cas } (-\beta) \text{ cas } \gamma \\ + & \text{ cas } \alpha \text{ cas } \beta \text{ cas } (-\gamma) \\ - & \text{ cas } (-\alpha) \text{ cas } (-\beta) \text{ cas } (-\gamma) \end{aligned} \quad (11)$$

This can be used to reduce the amount of operations necessary for the combination of the casine-terms in multi-dimensional HTs.

4 Applications of the Hartley Transform in Volume Rendering

The field of applications for the HT is very broad. In general, it can be applied to problems that deal with fluctuating phenomena, like for example in plasma physics, semiconductor physics, microwave acoustics, oceanography or radar mapping [5].

However, we are primarily interested in its applicability in volume rendering for medical imaging purposes where the HT is useful for, e.g., filter analysis for reconstruction purposes or frequency domain volume rendering.

4.1 Filter Analysis

In medical imaging one usually deals with sampled data sets obtained from acquisition devices like CT (computed tomography) scanners. Consequently, a fundamental operation in volume rendering is the reconstruction of a function from a set of samples, since most volume rendering algorithms require a resampling step and the resampling points usually do not coincide with the sampling points. Specifically, to obtain values of the function in between sample points some kind of interpolation has to be performed.

Schafer and Rabiner [20] state that in a frequency domain interpretation of interpolation it is clear that "interpolation is fundamentally a linear filtering process". In other words, interpolation is convolution of the sampled data with some filter. Provided that the sampled function is bandlimited (i.e., it only contains frequencies different from zero within a certain interval $[-\omega_m; \omega_m]$) and that it was sampled properly (according to Shannons sampling theorem [21]) the function can be reconstructed exactly from its samples with the filter [18]

$$\text{sinc}(x) = \begin{cases} \frac{\sin \pi x}{\pi x} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases} \quad (12)$$

This filter has infinite spatial extend, so some finite approximation has to be used in practice, which will not, however, reconstruct the original function exactly. A quite common method to assess the quality of reconstruction filters is to compare the frequency

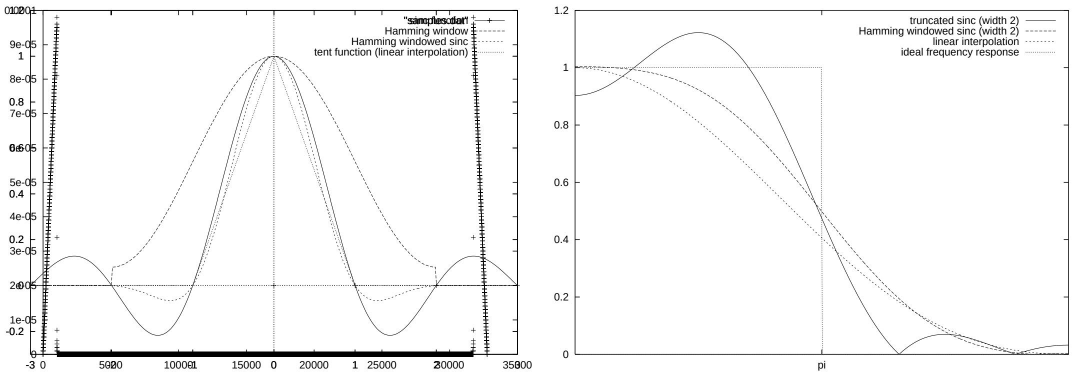

Figure 2: On the left several function reconstruction filters are depicted: the ideal function reconstruction filter, the sinc function, the Hamming windowed sinc function (where the Hamming window itself is also depicted) and the tent function, which corresponds to linear interpolation. On the right the corresponding frequency responses are depicted.

response of the filter to the frequency response of the ideal reconstruction filter (which is the box function for the ideal function reconstruction filter). This approach has been widely used in computer graphics [17, 13] and volume rendering [1, 9, 16].

Some exemplary function reconstruction filters are depicted in Fig. 4.1 on the left side, on the right side the corresponding frequency responses are depicted, calculated with the algorithm presented in Section 2.3 and Fig. 2.3. Obviously, the practical reconstruction filters only crudely approximate the desired frequency response of the ideal function reconstruction filter so that smoothing will be introduced as certain frequencies are attenuated, and aliasing will occur as higher frequencies are not completely suppressed.

4.2 Volume Rendering Based on the Projection-Slice Theorem

An algorithm which makes use of both the two- and three-dimensional HT is frequency domain volume rendering (FDVR), proposed independently by Dunne et al. [8] and Malzbender [15]. FDVR computes projection images of a three-dimensional data set in $O(n^2 \log n)$ time, after an $O(n^3 \log n)$ preprocessing step. This is achieved by exploiting the projection-slice theorem, which states, in the 3D case, that a projection of a volume can be computed by taking the (two-dimensional) inverse Hartley (or Fourier) transform of a slice, passing through the origin, of the 3D transform of the volume.

The algorithm mainly consists of three steps:

1. Transform the volume data to frequency domain (preprocessing).
2. Extract a slice from frequency spectrum which passes through the origin with the normal parallel to the viewing direction.
3. Back-transform the (two-dimensional) slice.

Obviously, the costly three-dimensional transform has to be performed only once per data set. When the viewing direction is changed only another slice must be extracted from the frequency volume and back-transformed. This transformation is only two-dimensional and reduces the overall computational cost of the algorithm considerably.

However, this approach suffers from some problems. First, since the projection obtained by FDVR is a line integral normal to the viewing direction (in other words, voxels on one viewing ray contribute equally to the result regardless of their distance to the viewer), the resulting images lack occlusion and often look like X-rays of the data set [15].

This can be overcome though. Levoy notes [14], that occlusion is not the only cue for the human visual system to recognize shapes of and relationships between objects. Spatial preprocessing can be used to provide depth cueing and directional shading, which is, however, quite memory demanding since several copies of the data set have to be kept in memory [14]. As data sets in volume rendering tend to be be quite big this is intolerable. Consequently, Totsuka and Levoy developed a method which implements depth cueing by frequency domain differentiation and directional shading by frequency domain multiplication [23].

High memory demand of FDVR, the second problem, however remains since more complex arithmetics in frequency domain demand a higher precision of the data set [23].

Nevertheless, FDVR proves to be quite a useful method for volume rendering. Most of the loss of realism due to lack of occlusion can be restored by depth cueing and directional shading and there is an undeniable benefit in speed. In Fig. 4.2 two examples of the output of this algorithm are given. A projection of a CT (computed tomography) data set of a human head is depicted on the left and a projection of a MRI (magnetic resonance imaging) data set of a human kidney is depicted on the right.

5 Conclusion

We have presented an easily comprehensible implementation of the HT and its multi-dimensional extension. This implementation has been shown to be easily adaptable to some important applications in volume rendering, e.g., frequency domain volume rendering or analysis of reconstruction filters by comparing the frequency responses to the ideal one.

We advocate the use of the HT because it has several advantages over the FT. It is real valued only whereas the FT will, in general, yield a complex result, and it offers a speed advantage compared to the FT on real valued data sets (and data sets used in volume rendering usually are real). Besides, it is its own inverse which further simplifies its use.

Acknowledgments

Special thanks for discussions about this paper and valuable hints go to Alexander Wilkie and for proof reading to Helwig Löffelmann. The data sets of the human head and the human kidney are courtesy V^is^M^ed^ project. Please refer to `http://www.vismed.at` for further information.

References

[1] Mark J. Bentum, Barthold B. A. Lichtenbelt, and Tom Malzbender. Frequency Analysis of Gradient Estimators in Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242–254, September 1996.

[2] James F. Blinn. Jim Blinn's corner — what we need around here is more aliasing. *IEEE Computer Graphics and Applications*, 9(1):75–77, January 1989.

[3] James F. Blinn. Return of the jaggy. *IEEE Computer Graphics and Applications*, 9(2):82–89, March 1989.

[4] Ronald N. Bracewell. *The Hartley Transform*. Oxford University Press, Walton Street, Oxford OX2 6DP, UK, 1986.

[5] Ronald N. Bracewell. The Fourier transform. *Scientific American*, pages 62–69, June 1989.

[6] Ronald N. Bracewell. Assessing the Hartley transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(12):2174, l99O.

[7] Ronald N. Bracewell, Oscar Buneman, Hong Hao, and J. Villasenor. Fast two-dimensional Hartley transform. *Proceedings of the IEEE*, 74:1282–1283, 1986.

[8] Shane Dunne, Sandy Napel, and Brian Rutt. Fast reprojection of volume data. In *Proceedings of the First Conference on Visualization in Biomedical Computing*, pages 11–18, May 1990.

[9] Michael E. Goss. An adjustable gradient filter for volume visualization image enhancement. In *Proceedings of Graphics Interface '94*, pages 67–74, Banff, Alberta, Canada, May 1994. Canadian Information Processing Society.

[10] Hong Hao and Ronald N. Bracewell. A three-dimensional DFT algorithm using the fast Hartley transform. *Proceedings of the IEEE*, 75(2):264–266, 1987.

[11] Ralph V. L. Hartley. A more symmetrical Fourier analysis applied to transmission problems. *Proceedings of the IRE*, 30:144–150, March 1942.

[12] Emmanuel C. Ifeachor and Barrie W. Jervis. *Digital Signal Processing: A Practical Approach*. Addison-Wesley, 1993.

[13] Robert V. Kenyon J. Anthony Parker and Donald E. Troxel. Comparison of interpolating methods for image resampling. *IEEE Trans. on Medical Imaging*, MI-2(1), March 1983.

[14] Marc Levoy. Volume rendering using the Fourier projection-slice theorem. In *Proceedings of Graphics Interface '92*, pages 61–69, May 1992.

[15] Tom Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, July 1993.

[16] Stephen R. Marschner and Richard J. Lobb. An evaluation of reconstruction filters for volume rendering. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings of the Conference on Visualization*, pages 100–107, Los Alamitos, CA, USA, October 1994. IEEE Computer Society Press.

[17] Don P. Mitchell and Aru N. Netravali. Reconstruction filters in computer graphics. *Computer Graphics*, 22(4):221–228, August 1988.

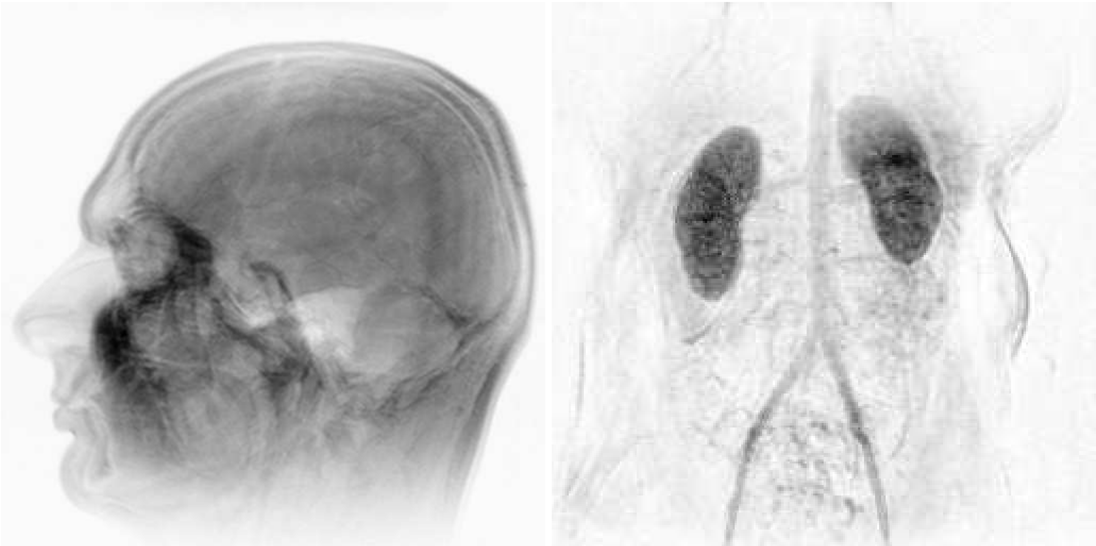

Figure 3: Two images generated with frequency domain volume rendering: a CT data set of a human head on the left and a MRI data set of a human kidney on the right.

[18] Alan V. Oppenheim and Roland W. Schafer. *Digital Signal Processing*. Prentice Hall, Englewood Cliffs, 1975.

[19] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge UP, 1988.

[20] Ronald W. Schafer and Lawrence R. Rabiner. A digital signal processing approach to interpolation. *Proc. IEEE*, 61(6):692–702, June 1973.

[21] Claude E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37:10–21, January 1949.

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

[23] Takashi Totsuka and Marc Levoy. Frequency domain volume rendering. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 271–278, August 1993.