PACIFIC GRAPHICS CONFERENCE 2002 Proceedings / pp. 116–126 Sabine Coquillart, Heung–Yeung Shum, Shi–Min Hu (eds.)

The Light Field Oracle

Reto Lütolf, Bernt Schiele, Markus H. Gross

Department of Computer Science,
ETH Zurich, Switzerland.
{luetolf, schiele, grossm}@inf.ethz.ch

Abstract

We present the light field oracle, a novel mathematical concept for the acquisition, processing and representation of light fields. We first compute a hierarchical representation from a set of sparse image samples using a combination of wavelet transform and scattered data interpolation. The light field oracle then progressively acquires image data and selectively refines this initial representation. By comparing the actual input image to the corresponding reconstruction from the wavelet pyramid, the oracle dynamically decides on whether the new sample is needed and, if necessary, inserts it into the representation. Our incremental update scheme exploits the spatial localization of wavelets and allows for highly efficient image decomposition. Likewise, image reconstruction for rendering is computed locally in the wavelet domain and does not require a global inverse transform. The wavelet hierarchy along with fast decomposition and rendering operators constitutes a powerful mathematical framework also amenable to compression.

1. Introduction

Image based rendering (IBR) has received a lot of attention since *Light Field Rendering* [13] and *The Lumigraph* [7] have been presented in 1996. The beauty of IBR lies in its power to represent scenes with rich and complex geometric detail. Hence, much of the follow–up work suggested various ways and tools to reconstruct efficiently approximations of the underlying *Plenoptic function* [1]—mostly from a set of dense scene samples and sometimes in combination with rough scene geometry. As a result, the initial concepts have been improved significantly in terms of storage efficiency and reconstruction quality.

In spite of a few exceptions, however, relatively little work has been devoted to the development of fast and easy–to–handle data acquisition schemes for light field rendering systems. In this paper, we present the light field oracle—a novel mathematical concept for progressive, hierarchical light field acquisition, representation, and processing. The representation combines conventional wavelet transforms with hierarchical scattered data interpolation and works with any orthogonal wavelet and interpolation filter kernel. In addition, we designed a set of local decomposition and reconstruction operators which greatly reduce the computational costs of incremental updates and rendering queries.

In IBR, in–core compression has proved to be of critical importance for rendering, especially when not involving geometric scene information. The challenge has been to use as little memory as possible during acquisition and reconstruction while still providing the necessary random access to every single data item. Besides an effective compression we could think of additional data reduction mechanisms, e.g. by intelligently selecting the data samples. The *Light Field Oracle* implements such a smart sampling and data selection scheme. The initial input to the oracle is a small, incomplete set of images of a 3D scene, sampled at irregular positions. The oracle's task then is to decide—automatically and progressively—whether or not to accept a new image sample that was not included in the initial set. Accepted images are incrementally inserted into the hierarchy, each of which incrementally refining the initial representation, rejected images are discarded. The hierarchical data representation itself as well as powerful view interpolation properties are provided by an extended 4-dimensional wavelet transform. This concept allows us to design highly efficient local projection operators for data insertion as well as local reconstruction operators for rendering. The hierarchical representation in combination with the local operators form the core contribution of this paper.

The implementation presented here is tested using synthetic, pre–computed reference images: A stack of ray-traced images of a 3D scene serves as source of input. However, our concept can easily be adapted to a real–world setting employing a hand–held camera as source of input, given a decent tracking of camera position and orientation.

In the following section we discuss related work. We then give a short overview of the entire system and its main components. In Section 4 we present the employed param-

eterization before we focus on the data representation in Section 5 and the local operators in Section 6. We demonstrate the performance of the method on various example data sets in Section 7, followed by the conclusions. Finally, we discuss work that is currently underway in Section 9.

2. Related work

The notion of hierarchy can be found in several approaches to light field representation and encoding. For instance, a 2–dimensional wavelet transform using Haar wavelets is applied on a linearized representation of the *Spherical Light Fields* in [10]. A very similar coding scheme for very large volume data has been introduced before by the same authors in the context of volume rendering [9]. Furthermore, fully 4-dimensional wavelet transforms are exploited by several authors: In [17], a 4–dimensional, non–standard Haar transform is utilized. The tree–like coding scheme called *Wavelet Stream* is designed for progressive storage and transmission of compressed light field data as well as for interactive rendering by providing random access to coefficients while decoding. Comparable work is presented in [15] and also in [12]. The authors of [12] additionally include detailed discussions about parameterization issues in their setting, the choice of decomposition for higher dimensional wavelet transforms as well as the choice of wavelet basis functions. However, unlike the light field oracle, all these approaches compute the hierarchy only after data acquisition rather than as an integral part of it.

Numerous solutions have also been proposed for the compression of light fields. A short but comprehensive overview may be found in [17]. A detailed comparison among block, reference and wavelet coders is given in [14]. It is well known that using even a rough approximation of the scene geometry—the *scene proxy* [2]—can greatly reduce the number of image samples needed. In purely image based rendering systems, however, efficient representation and compression is of greatest importance. The question of appropriate sampling rates for anti–aliased light field rendering, i.e. minimal *Plenoptic Sampling* rates, is addressed in [5]. These rates can be found by analyzing the bounds of the spectral support of the light field signal. However, the question how to acquire this minimal set of images has not been answered.

In [18], an adaptive acquisition method for lumigraphs is presented. While it includes geometrical scene information for improving the representation, it is confined to synthetic scenes only. Image data acquisition of real world scenes is described in [8]. A hand–held camera is used for video data capturing whereas a structure–from–motion approach is applied for image sequence calibration. Our approach nicely complements on these methods in that it provides a powerful framework for the underlying mathematical representation. Furthermore, our scheme is inherently progressive and efficiently refines the initial light field representation.

The *Unstructured Lumigraph Rendering* (ULR) algorithm [2] bridges between systems involving geometry and systems that do not by describing a generalized representation for many image–based rendering (IBR) algorithms. It also bridges between structured input as used in the original light field method and unstructured input first dealt with in [7]. Furthermore, the method presented in [11] introduces *dynamic reparameterization* as a means to deal with significant, unknown depth–variation without requiring approximate scene geometry.

3. Overview

Figure 1 depicts the conceptual components of the light field oracle. Besides acquisition, parameterization and the

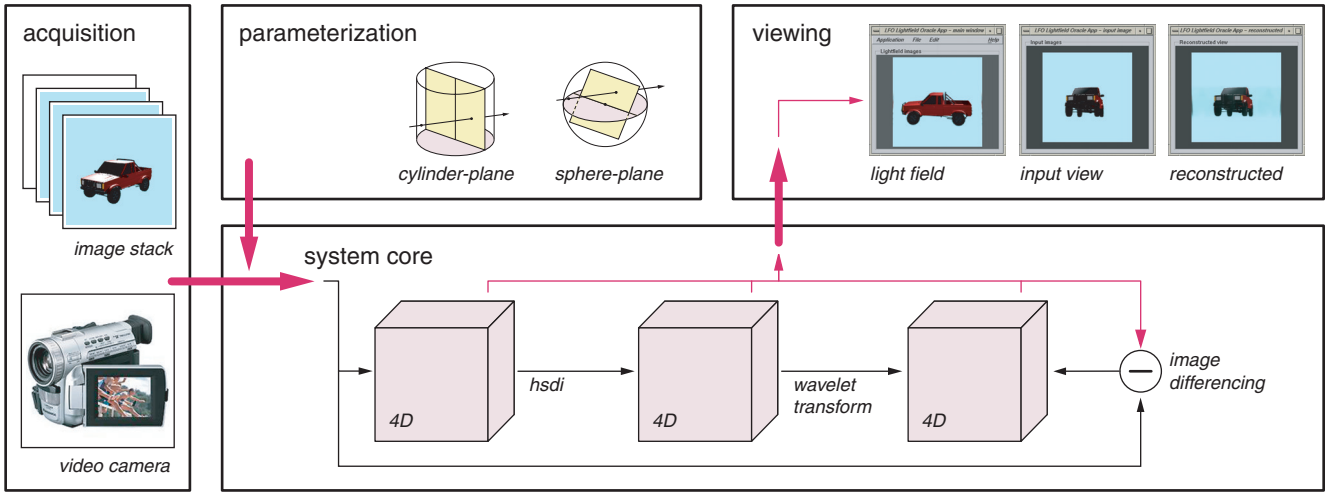

Figure 1: Overview of the setup: acquisition, parameterization, viewing and core component.

viewer, the core component is constituted by the hierarchical representation and by a set of operators.

The *acquisition component* provides input images combined with the corresponding position and orientation information of the real or virtual camera. In the current test implementation, it delivers individual images from a stack of previously raytraced images on demand instead of reading from a live video stream. Thus, raytraced images are available; they are not generated when needed as is done in [18].

Secondly, a *parameterization module* is needed to transform image and camera pose data into samples and their corresponding 4–dimensional indices required in the light field representation. We call a single sample a *hyxel* [1] each of which consists of the four components αRGB. The parameterization is discussed in Section 4.

The *core component* itself has two modes. In the first mode, the *approximation mode*, a light field estimate is computed by projecting an initial set of input images using a given parameterization, as it is done in [7]. A quadra–linear basis function is employed for this projection. The resulting light field is sampled only coarsely and may even contain undefined locations, especially when the set of input images is kept small. In order to improve on the estimate, a hierarchical scattered data interpolation (HSDI) based on the scheme in [7] is performed. However, unlike [7], we modify the hierarchical computation to both interpolate and build a wavelet representation. To this end, we designed an extended multiresolution analysis (eMRA) which is discussed in detail in Section 5.

Once an initial estimate is found, the core switches to its second mode, the so called *oracle mode*. In this mode, aiming at a progressive refinement of the estimate, the core continuously grabs image samples with new information. Before further processing new samples, the oracle decides on the acceptance of each of them individually based on an analysis of the difference between the input image and the corresponding reconstructed view from the current camera position. The required rendering procedure utilizes image slicing, i.e. extraction of a 2D image slice from the 4D data set. Accepted image samples are incrementally inserted as parameterized data into the hierarchy, rejected ones are discarded. Note that the oracle decides on each input image individually. Hence, the oracle does not aim at finding the optimal set of images representing a given 3D scene as a whole, but rather at locally minimizing the amount of data being acquired. As a result, the set of selected image samples will be dependent on the order in which the images are being presented.

[1] According to the *pixel* (picture element) in 2D and the *voxel* (volume element) in 3D we call an element in 4D a *hyxel* (hyper–volume element).

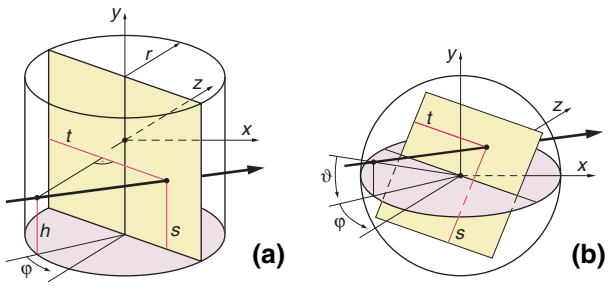

Figure 2: Cylinder–plane (a) and sphere–plane (b) parameterization.

In order to minimize the computational cost of the progressive refinement we introduce local projection operators which directly alter the extended MRA. For the purpose of rendering, we designed local reconstruction operators. Both, the local projection and the local extraction operators are addressed in detail in Section 6.

The *viewer component* provides all necessary functionality concerning viewing and controlling of the system. It mainly consists of a light field viewer and three special viewers for 1) the current input image, 2) the reconstructed image at the corresponding view point and 3) a 3D visualization of the spatial camera position and orientation of accepted and rejected input images in relation to the object's position. The latter viewer may be understood as some sort of user guidance system by visualizing already densely and still sparsely sampled locations around the object of interest.

4. Parameterization

We parameterize each viewing ray using a novel *cylinder–plane parameterization* that is similar to the sphere–plane parameterization presented in [4]. For every ray the intersection with a cylinder and a plane is computed. The cylinder circumscribes the object of interest centered at the origin. The plane itself is centered inside the cylinder, aligned parallel to the cylinder axis and orientated perpendicular to the current viewing direction as shown in Figure 2(a). The points of intersection, i.e. the viewing ray, can be identified through the four parameters (φ, h) and (s, t).

This parameterization has various advantages. Firstly, when no geometrical information for depth–correction is available, it is imperative to place the focal plane near the object's center such that the blurring artifacts for points outside the focal plane are not intensified. Secondly, the cylinder provides full horizontal coverage and thus, it is not necessary to glue together two or even more parameterization spaces for $360°$ data capturing, as needed for the 2–plane parameterization [7], [13].

Moreover, compared to the standard 2–plane parameterization, the cylinder–plane parameterization proved to be considerably better and generally more homogeneous in terms of the signal–to–noise ratio (SNR) on camera paths that are important in our setting. This is demonstrated in Figure 3 where the camera was moved horizontally on a circular path over a range of $180°$. The angle is set to $\varphi = 0$ where two *light slabs* [13] of the 2–plane parameterization are merged together at an angle of $90°$. As a result, the peak in the red curve is caused by the presence of two focal planes for the same view. The SNR was computed as

$$SNR = 20 \cdot \log\left(\frac{E_S}{E_N}\right)$$

with E_S and E_N denoting the signal energy of the noisy signal and the pure noise, respectively. The signal energy can be computed as the sum of squares of all discrete values, i.e. pixel intensities. Reconstructed views using either parameterization were used as noisy signals whereas the difference of a reconstructed view and the corresponding raytraced reference image was used as the noise itself.

However, a minor difficulty of the cylinder–plane parameterization might arise when moving the camera away from the $z = 0$ plane to an elevated position, but still pointing to the origin. As is visualized in Figure 10, foreshortening effects will occur due to the perspective projection of the image plane onto the *st*-plane. For ray–based insertion and/or extraction, this is not an issue since the perspective is automatically taken into account. In our case of image–slicing however, the perspective distortion must be accounted for. Such a correction can be accomplished by post–warping images reconstructed through slice extraction, and pre–warping images prior to projection, respectively. Both operations can be accomplished using texture hardware for image resampling.

Using a different parameterization such as the *sphere–plane parameterization* shown in Figure 2(b) foreshortening may be avoided. By having the plane centered in the sphere's origin and oriented perpendicular to the viewing direction, perspective distortion does not occur. Future work will experiment with the sphere–plane parameterization as illustrated in Figure 2(b).

5. Data representation

The data representation is a central part of the light field oracle and crucial for its success. Most importantly, the data representation has to handle an incompletely sampled function. Furthermore, we have to manage adaptive sampling densities since samples will be selected randomly. It is also highly desirable that the multiresolution analysis is kept independent of the particular filter kernel employed

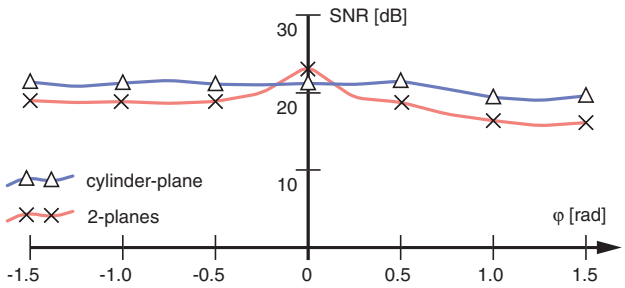

Figure 3: Comparison of the signal–to–noise ratio (SNR) of different light field parameterizations.

since the approximation of the light field is significantly influenced by the quality of the interpolation filter. Moreover, support for a progressive and efficient compression scheme with high compression gain and random access to every data element is important. Finally, we need fast local operators for data insertion and reconstruction.

In order to meet all these requirements we designed an *extended multiresolution analysis* (eMRA) as a unified representation for sampling, interpolation, compression and rendering. As in [17] we apply a fully 4–dimensional wavelet transform with the additional freedom of an independent projection/interpolation filter kernel as explained below. For a comprehensive introduction to wavelet theory we refer to [19] and [6]. As pointed out earlier, the core module has two modes: an initial construction of a light field estimate during the *approximation mode* (Section 5.1) as well as the *oracle mode* for incremental refinement of the initial estimate (Section 5.2).

5.1. Construction of the extended MRA

Figure 4 shows the initial projection path, the subdivision and fill–in procedure as well as the projection onto difference space. We use 1D notation for better readability although we deal with a 4-dimensional set of hyxels. In essence, a hierarchical scattered data interpolation similar to the splat–pull–push scheme in [7] is applied to an initial set of parameterized image data samples. The splat–pull–push scheme combines methods to address the problems of sparse sampling with large gaps of missing data [3] and methods to deal with the non–uniformity of the sample distribution [16]. Gaps of missing data will be denominated with NO_DATA, as is indicated in the initial data vector $\boldsymbol{c}^0$ in Figure 4, for instance.

The major difference in comparison with the splat–pull–push scheme is that we convert the hierarchy into a consistent MRA while constructing it. To this end, we first push the data down the hierarchy. Note that the corresponding projection operator P^{m^*} at level m can be any interpolation filter, e.g. a Gaussian or a B-spline, modified to recognize and handle locations with missing data. In other words, this

operator is independent of the operator setting employed in the wavelet transform. Subsequently, data is pulled up again level by level using the wavelet subdivision operator S^m. During subdivision, an additional fill–in is performed that eventually restores previously known data in each frequency subband. The corresponding operator R^m computes the data $\bar{\boldsymbol{c}}^m$ at level m from the estimate $\tilde{\boldsymbol{c}}^m$ and from the initial projection $\boldsymbol{c}^m$ as shown in Equation (1). Note that R is non-linear.

$$\bar{\boldsymbol{c}}^m = R^m(\tilde{\boldsymbol{c}}^m, \boldsymbol{c}^m) \qquad \text{with}$$
$$\bar{c}_j^m = \begin{cases} \tilde{c}_j^m, \text{ if } c_j^m = \text{NO\_DATA and } m < M \\ c_j^m, \text{ otherwise} \end{cases} \tag{1}$$

The non–linearity of the setting requires to propagate the changes down the pyramid to all $\bar{\boldsymbol{c}}^n$, $n > m$ each time a new $\bar{\boldsymbol{c}}^m$ is computed. This is accomplished by standard wavelet projection operators P^m and Q^m.

Figure 11 demonstrates the power of the hierarchical interpolation scheme in a 2D setting. Image (a) shows the original image. Image (b) shows a scattered version of the original, sampled on randomly oriented lines such that only 25% of the input data is known. The interpolated image (c) was generated using a 5–tap Gaussian. When applying the scheme in 4D, interpolation is not only done in individual image slices but also across neighboring slices.

The initial data vector $\boldsymbol{c}^0$ is computed by projecting the discrete light field function f onto a chosen basis. This is accomplished by the inner products of f with the duals $\tilde{\phi}$ of the basis functions as shown in Equation (2).

$$c_j^0 = \langle f, \tilde{\phi}_j^0 \rangle = \int f(x) \cdot \tilde{\phi}_j^0(x)\, dx \qquad \text{with} \tag{2}$$

$$f(x) = \sum_{i=1}^{k} f_i \cdot \delta(x - x_i) \qquad \delta\text{: Kronecker Delta} \tag{3}$$

$f(x)$ is given by the set of k scattered samples at some spatial positions x_i within the support of ϕ. Inserting (3) in (2) yields

$$c_j^0 = \langle \sum_{i=1}^{k} f_i \cdot \delta(x - x_i), \tilde{\phi}_j^0(x) \rangle = \sum_{i=1}^{k} f_i \langle \delta(x - x_i), \tilde{\phi}_j^0(x) \rangle = \sum_{i=1}^{k} f_i \cdot \tilde{\phi}_j^0(x_i)$$

In practice, we use samples of the primal functions ϕ instead of the duals $\tilde{\phi}$. In the case of the box basis, $\tilde{\phi} = \phi$ applies. The duals of the quadra-linear basis—the basis we are using for the initial projection—are more complex, but the basis functions sufficiently approximate their own duals for our purposes, assuming nearly orthogonal decompositions.

5.2. Incremental update and oracle mechanism

After the initial construction of the pyramid we switch to local operators which involve only the subset of wavelet coefficients actually affected by an operation. Local operators for both update and rendering procedures are advantageous for the reduction of the computational complexity. They are discussed in detail in the next section.

An incremental update of the hierarchical representation works as follows. Let $\bar{\boldsymbol{c}}_{new}^0$ be a new input image. We push the new data item down the hierarchy exploiting the linearity of the projection operator. This results in the recurrence relation expressed in Equation (4), where $\Delta\bar{\boldsymbol{c}}^m$ denotes the increment with respect to $\bar{\boldsymbol{c}}^m$:

$$\begin{aligned} \bar{\boldsymbol{c}}_{new}^m &= P^{m-1}\, \bar{\boldsymbol{c}}_{new}^{m-1} = P^{m-1}(\bar{\boldsymbol{c}}^{m-1} + \Delta\bar{\boldsymbol{c}}^{m-1}) \\ &= \bar{\boldsymbol{c}}^m + P^{m-1}\Delta\bar{\boldsymbol{c}}^{m-1} = \bar{\boldsymbol{c}}^m + \Delta\bar{\boldsymbol{c}}^m \qquad m > 0 \end{aligned} \tag{4}$$

Thus, it is only the incremental information $\Delta\bar{\boldsymbol{c}}^m$ that needs to be inserted. Note that this increment represents local changes only.

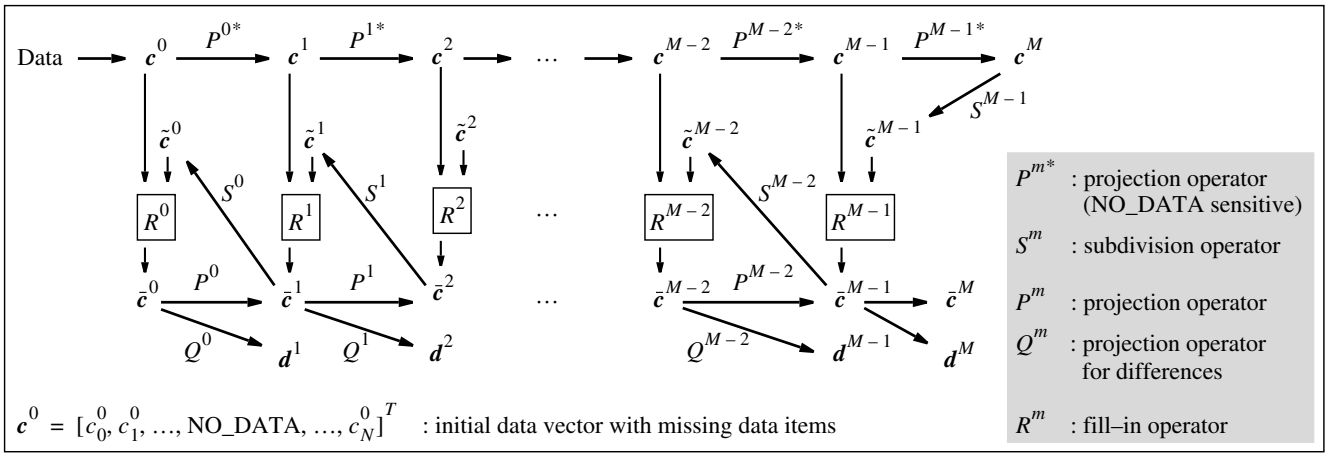

Figure 4: Extended MRA used to construct the hierarchical light field representation.

The oracle mechanism itself is rather simple. The core module renders a view that corresponds to the position of the new image data sample and computes the difference of input and estimated view. Any difference represents missing information. If the difference fulfills a threshold criterion it is fed into the eMRA using a local, incremental insert operator. With $\bar{c}^0_{new}$ being a new image data sample and $\bar{c}^0$ the corresponding rendered view, the differences $\Delta\bar{c}^0 = \bar{c}^0_{new} - \bar{c}^0$ are decomposed using the insert procedure shown in Equation (4). Of course, the same scheme is applicable to get $\boldsymbol{d}^m_{new}$ by using Q^{m-1} instead of P^{m-1}.

6. Local operators

As pointed out before, an important contribution of the paper is to minimize the computational complexity of the incremental updates by the use of local operators. In general, the complexity of a wavelet transform for an n-dimensional data set of size N in each dimension is of $O(N^n)$, which is $O(N^4)$ in our case. Clearly, we need to avoid a complete back–transform with subsequent re–transform each time an update is to be performed. Consequently, we constructed a local projection operator used for incremental updates as well as a local reconstruction operator used for rendering. Since these operators basically work on 2D slices through the 4D data set, i.e. images, they are of $O(N^2)$ as discussed more precisely in the results section.

The *progressive inverse wavelet synthesis* (PIWS) [20] represents a somewhat similar idea. The scheme allows for random access and partial decoding of wavelet compressed 3D data, based on inverse lifting operations. Unlike our local operators, the PIWS scheme is designed for reconstruction, i.e. rendering, only.

6.1. Concept

In order to build local operators for both projection and reconstruction it is necessary to identify the affected subset of coefficients. The wavelet functions' properties of being well localized both in frequency and time—in contrast to the Fourier transform—allow us to do so. Due to the construction of the basis functions using binary dilations and dyadic translations of some single function ψ [6], it is possible to calculate the region of support, given a specific wavelet and an arbitrary position in the data field to be filtered. This so–called *active area* is of the same dimensionality as the data field itself. Hence, the active region on level m of the decomposition pyramid for a filter ψ of size s centered at position j_ψ in 1D is bounded by

$$\textit{active area} = \left[j_{\psi,m} - \left(\frac{s}{2}\right)^{m-1}, j_{\psi,m} + \left(\frac{s}{2}\right)^{m-1} \right] \qquad (5)$$

where $j_{\psi,m}$ is the position of j_ψ on level m given by

$$j_{\psi,m} = \frac{j_\psi}{(2^m)}$$

The bounds defined in (5) may slightly vary depending on a specific filter's tap–size and how it is centered. In addition, boundaries must be handled separately in case the active region does not fully lie inside the data field. Our implementation solves this problem by following the wrap–around strategy. Hence, active regions may continue at the opposite end of the data field.

Identifying these active regions depending on the type of edit operation, e.g. update or extraction of a point, line, etc., and depending on the dimensionality of the data field allows for the design of local operators. Consequently, these operators access only the relevant subset of all wavelet coefficients affected by the operation and leave all others untouched.

Due to the separability of the wavelet transform, a multi–dimensional data field can be filtered following the tensor product strategy, i.e. all dimensions can be transformed independently, one after another.

For illustration, Figure 12 shows the active regions for a local point update for a level-3 transformed data field in 2D. Note that although the wavelet transform is applied separately to each channel of the input image they are merged together again for visualization. The two rightmost images show transformed representations of the left image, using Daubechies–4 (middle) and Daubechies–8 wavelets (right). The middle image visualizes the active regions for a local update operation on all 3 levels for the yellow dot located inside the yellow circle in the leftmost image. The active areas in all frequency subbands on the first two levels are clearly identifiable. The third level shows a wrap–around situation in both the x- and y-direction. The rightmost image, on the other hand, shows the active regions for a local update operation for the red dot inside the red circle in the original image. The regions' sizes are noticeable bigger compared to the middle image due to the double tap–size of the Daubechies–8 wavelet in comparison to the Daubechies–4 wavelet. Moreover, a wrap–around situation in both directions again occurs on level 2. On level 3, our algorithm automatically switches from local operators to global filtering since the active areas cover the whole relevant area.

6.2. Local reconstruction operator in 4D

By placing the camera position at a sample location (φ, h) on the cylinder, each image represents a 2D slice of the 4D data set for fixed (φ, h) values. For ease of notation, we will replace φ by x and h by y. Furthermore, the *st*-plane will be identified as the *zt*-plane. Hence, rendering a new image means reconstructing a complete *zt*-plane for a fixed

pair of (x, y) coordinates. Quadra–linear filtering according to the chosen projection basis can be accomplished through reconstruction of the image slices corresponding to the four neighboring xy sample locations of the desired view point and blending them together using accumulation buffer support of the graphics hardware.

When reconstructing a complete zt-plane, i.e. one image, global filtering in both z- and t-direction must be applied. Hence, active areas cannot be identified in these two dimensions; they can only be recognized in the x and y dimension for a given, fixed sample location (x_0, y_0). Thus, the local reconstruction operator cycles through the following steps, starting from the highest transformation level M:

```
(1) set current level to m = M;
(2) identify active areas in x and y,
    according to the position of (x0,y0)
    on the current level m;
(3) successively apply reconstruction filter
    in x- and y-direction locally and in z-
    and t-direction globally;
(4) set new current level to m = m-1;
(5) if (m <> 0) go back to step (2);
```

Figure 5 shows the above procedure for the last filtering step from level 1 to level 0 for the sample location (x_0, y_0). The zt-axis represents both the z and t dimension. Hence, a line aligned to this axis corresponds to a zt-plane. It is important to note that after filtering and downsampling, an active area for local filtering in any dimension collapses to the fixed, single coordinate at which the active area was centered, as is shown in Figure 5 for both x_0 and y_0.

Due to the collapsing of locally filtered dimensions, it is—in terms of computation time—crucial to first apply the reconstruction filter in such a dimension. As is clearly visualized in Figure 5, only one single zt-plane is left for further processing after filtering in x and y has been completed. Note that this applies only to the last step from level $m = 1$ back to $m = 0$.

6.3. Local projection operator in 4D

Whenever the oracle decides to insert new data into the hierarchy according to Equation (4) in Section 5.2, it is a difference image, i.e. a single zt-plane representing this difference image, that needs to be inserted incrementally into the eMRA. This plane is interpreted as a second 4D data set

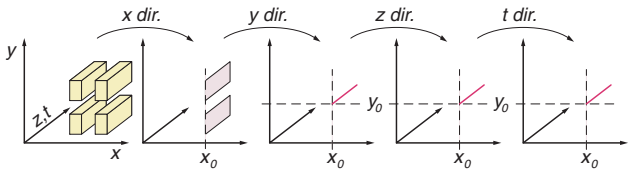

Figure 5: Filtering steps for a local reconstruction of a *zt*-plane in 4D, when filtering from level 1 to 0 at location (x_0, y_0).

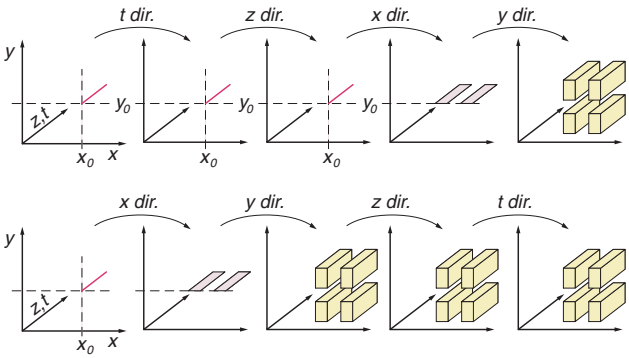

Figure 6: Filtering steps for a local projection of a *zt*-plane in 4D, when filtering from level 0 to 1 at location (x_0, y_0): optimal (top) and more costly order of filtering (bottom).

holding only one image sample. The projection of this single plane for a given, fixed location (x_0, y_0) works very similar to the reconstruction procedure given in the previous section, as is shown below:

```
(1) set current level to m = 0;
(2) identify active areas in x and y,
    according to the position of (x0,y0)
    on the current level m;
(3) successively apply projection filter
    in x- and y-direction locally and in z-
    and t-direction globally;
(4) set new current level to m = m+1;
(5) if (m <> M) go back to step (2);
```

Figure 6 shows this projection procedure for the first filtering step from level 0 to level 1 for the location (x_0, y_0). Note that contrary to the local reconstruction operator, the local projection does not collapse any active area. Instead, it inflates active areas from one level to the next as is already shown in 2D in Figure 12. Furthermore, active areas do not actually exist until the first step from level 0 to level 1 has been completed, as is demonstrated in Figure 6. Active areas in any dimension get generated after filtering in this dimension has been completed.

However, there is more than one possibility of how to complete one level. The top row of Figure 6 shows the optimal order, the bottom row shows a correct yet far more costly sequence. It is obvious that after the x- and y-directions have been completed, many zt-planes instead of just one need to be processed. Note that reordering of individual filtering steps at any level of the pyramid is allowed as long as each step is fully completed.

Figure 7 illustrates the impact of a chosen order of the individual filtering steps in each dimension on the computational cost in a 2D setting. An axis–aligned line is being transformed to level 1 using a local projection operator. Whereas in the bottom row the second step involves pro-

cessing of a whole set of lines, it is only one single line for the optimal order, as is shown in the top row.

A brief analysis of the computational cost additionally emphasizes the importance of finding the optimal sequence for the first transformation to level 1: When assigning the virtual cost of c for filtering an initially untransformed data field in 1D, we get a total cost of $2 \cdot c^2$ for the transformation of a 2D data field to level 1. Due to subsampling of each dimension we get a total cost of $2 \cdot (c/2)^2$ for the projection to level 2. On level 3, we get $2 \cdot (c/4)^2$, and so on. Summing up, we get the geometric series shown in Equation (6),

$$2c^2 + 2 \cdot \left(\frac{c^2}{4}\right) + 2 \cdot \left(\frac{c^2}{16}\right) + \ldots = 2c^2 \cdot \sum_{m=0}^{M} \frac{1}{(2^{2m})} \qquad (6)$$

where m denotes again the current level of transformation. In an n-dimensional space and using the properties of geometric series, we get

$$nc^n \cdot \sum_{m=0}^{M} \frac{1}{(2^{n \cdot m})} \rightarrow \frac{nc^n}{1-q} \quad \text{with } q = \frac{1}{2^n} \quad \text{for } M \rightarrow \infty$$

For $n = 1$, the transformation to level 1 is already half of the cost of all following steps. In 2D, the first step already dominates the overall cost. Thus, the higher the dimensionality, the more we gain by efficiently handling the projection to level 1.

7. Results

The concepts presented in the previous sections have been implemented and tested on various input data sets. We first discuss the visual performance of the light field oracle on two different sets of input images. We then comment on results demonstrating the performance in terms of computation time.

Figure 13 shows the visual performance of our concept of local, incremental projections. Image (a) is a reconstructed view of a densely sampled light field, rendered from the transformed data set using local reconstruction operators. The light fields' resolution was set to $16^2 \cdot 256^2$ and it was constructed from initially 250 input images, randomly selected from a stack of 450 pre–rendered images. The transformation was completed to level 1 using the Haar basis. Image (b) is a raytrace of the same view as for image (a). Note the differences in high frequencies between those two images. We take the difference image shown in (d) to perform an incremental update according to Equation (4). We use the local projection operator and again render the same view using the local reconstruction operator which yields image (c). The resulting image is hardly distinguishable from the input image (b). Note that,

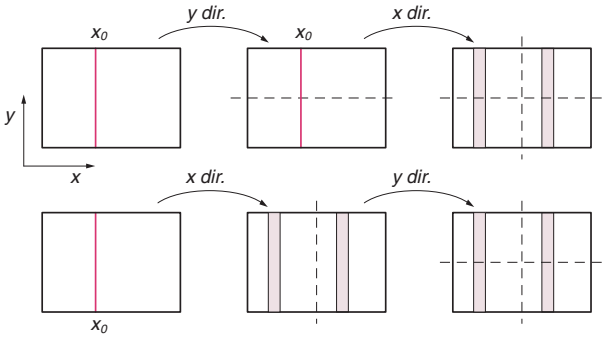

Figure 7: Local projection of an axis–aligned line in 2D, projecting to level 1: optimal (top) and more costly order of filtering (bottom).

in general, a lumigraph cannot reproduce its input images [2].

Figure 14 demonstrates all processing steps and both modes of the light field oracle. Image (a) shows a rendering right after the initial construction of the light field. The set of pre–rendered images for this light field contains 150 images in total. They are regularly distributed on a grid of 30 (horizontal) times 5 (vertical) sample locations, resulting in a coarse sampling in the vertical direction. 120 images were randomly selected for the construction of the initial light field of resolution $16^2 \cdot 256^2$. The black regions in the image show areas with missing data. Image (a) was rendered using ray–based extraction. Image (b) shows the result after the transformation of the light field into our extended MRA as described in Section 5, seen from the same viewpoint. The transformation was completed to level 1 using the Haar basis. Note the ghosting artifacts of neighboring views as a result of the hierarchical interpolation procedure. Again, image (b) was rendered using the local reconstruction operators. Images (c)–(e) demonstrate the oracle mode, as already shown in Figure 13. Using the difference image (e) for an incremental update according to Equation (4) and again rendering the same view using the local reconstruction operator results in image (d).

As stated in Section 6, the local operators are of $O(N^2)$ instead of $O(N^4)$ since they basically work on images instead of the 4-dimensional data set as a whole. A more thorough analysis of the computational complexity yields a total dependence of

$$O(M) \cdot O(w) \cdot O(N^2) \qquad (7)$$

where N denotes the size of those dimensions that need to be filtered globally. According to Section 6 these are the z and t coordinates and match with the image resolution. The first term $O(M)$ —with M being the maximal level of transformation—accounts for the depth of the transformation

pyramid whereas the second term $O(w)$ considers the width of the wavelet filter.

As stated before, most of the computational work needs to be done during the transform to level 1. Thus, the overall complexity shows a weak dependence of the first term in Expression (7), as is demonstrated in Figure 8. These timings were measured running our implementation on a single MIPS R12k processor at 400 MHz. The left column shows the timings measured for a data set transformed to the maximal possible level for the Haar and the Daubechies–6 wavelets. Each chart displays the time spent for a full wavelet transform of the whole data set, for a local reconstruction and for a local projection operation, dependent on the resolution in x and y. The size of z and t were both fixed at 64. The right column shows the values for the same setting when only transforming to level 1 instead of the maximal possible depth. Firstly, the quadratic increase in computation time for the global decomposition is clearly recognizable. Secondly, the curves of the local operators do not exhibit a similar increase in computation time. In case of a wavelet transform to the maximal

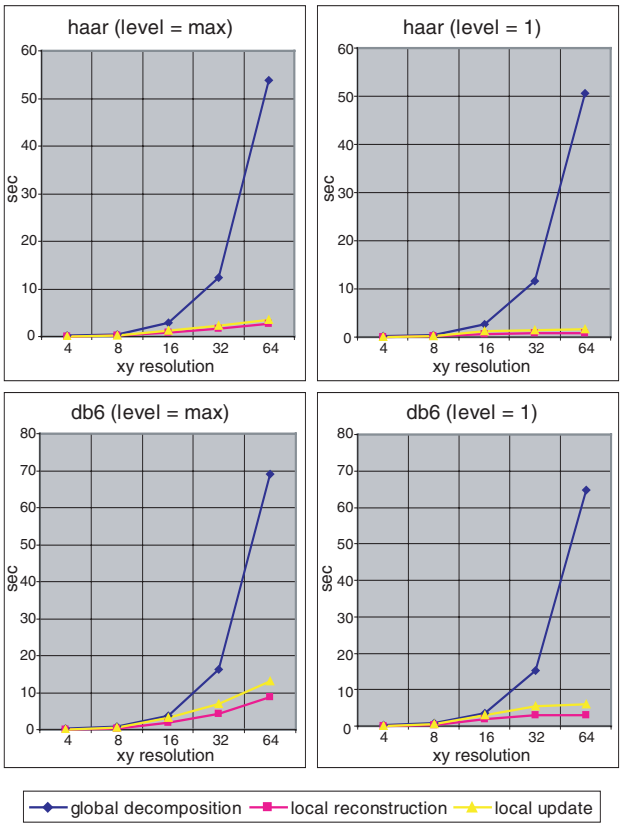

Figure 8: Timings measured for different wavelets dependent on the resolution in *x* and *y*: maximal possible depth of transformation (left column) and transformation to level 1 (right column). The size of *z* and *t* is set to 64.

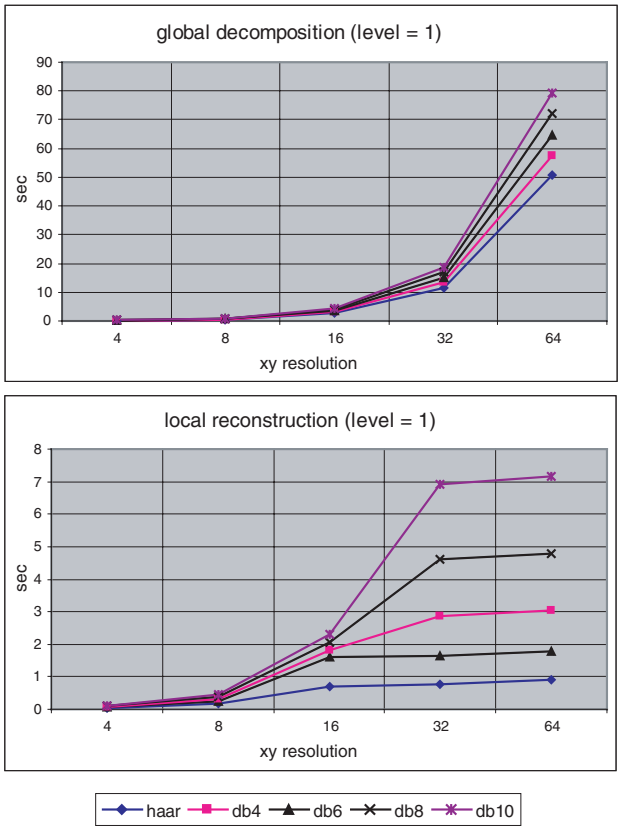

Figure 9: Timing charts comparing global decomposition with the local reconstruction operator for *z* = *t* = 64.

possible depth of the pyramid (left column) they behave linearly which is reflected by the first term in Expression (7). In case of a level-1 transform only, the curves start to level off to near constant behavior at a certain point, as is additionally emphasized in the bottom chart of Figure 9. This point can be identified as the size of x and y where active areas for local filtering for a given wavelet can be computed. Once this point is reached, the *computation time is independent of the size of x and y* as is stated in Expression (7).

As already mentioned, Figure 9 (bottom) shows the xy values at which the computation time for the local operators levels off to nearly constant behavior. The precise values are dependent on the tap–size of the respective wavelet, as indicated by the second term in Expression (7). To give an example, the Daubechies–6 wavelet starts showing a constant behavior earlier than the Daubechies–10 wavelet as a consequence of its smaller tap–size. For comparison, the upper chart in Figure 9 gives the timings measured for the full decomposition. The size of z and t were again both fixed at 64. The transforms for each wavelet were completed up to level 1.

Our implementation accepts any orthogonal wavelet. Moreover, the hierarchical scattered data interpolator works with any reasonable interpolation filter.

8. Conclusions

We presented the light field oracle—a concept for progressive data acquisition and representation of light fields, exploiting the benefits of a hierarchical scattered data interpolation scheme and wavelet coding, combined in an extended MRA. Moreover, we introduced local operators for both projection and reconstruction needed for the progressive refinement.

The results presented demonstrate the great potential for a future real–world system using a hand–held camera. The timing measurements show that interactive frame rates for both local reconstruction and local projection operations are feasible, as soon as the system runs on one of the most recent GHz processors: Using a Daubechies–4 wavelet for example, the time needed for a local reconstruction operation will come down to somewhat less than half a second. Additional future work on smart caching mechanisms will allow for just–in–time (*JIT*) rendering [20].

9. Future work

Besides caching mechanisms, we will first of all address compression by exploiting the potential for high compression gains of the wavelet transform in the near future. First experiments with a custom data structure capable of storing only the non-zero coefficients while still permitting random access to arbitrary values—as needed by the presented algorithms—show that compression ratios of 100:1 or more are achievable, as is reported in [14], for instance.

In addition, we will include the sphere–plane parameterization as is shown in Figure 2(b). In a real–world setting, camera calibration and tracking will become important issues to address.

References

[1] E. Adelson and J. Bergen. “The plenoptic function and the elements of early vision.” In M. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, MA, 1991.

[2] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. “Unstructured lumigraph rendering.” In E. Fiume, editor, *SIGGRPAH 2001 Conference Proceedings*, Annual Conference Series, pages 425–432. ACM SIGGRAPH, Addison Wesley, 2001.

[3] P. J. Burt. “Moment images, polynomial fit filters, and the problem of surface interpolation.” In *Proceedings of Computer Vision and Pattern Recognition*, pages 144–152, 1988.

[4] E. Camahort, A. Lerios, and D. Fussel. “Uniformly sampled light fields.” In G. Drettakis and N. Max, editors, *Proceedings of the 9th Eurographics Workshop on Rendering*, pages 117–130. Eurographics, Springer Wien, 1998.

[5] J. Chai, X. Tong, S. Chan, and H. Shum. “Plenoptic sampling.” In K. Akeley, editor, *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 307–318. ACM SIGGRAPH, Addison Wesley, 2000.

[6] C. Chui. *An Introduction to Wavelets*. Academic Press, 1992.

[7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. “The lumigraph.” In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, 1996.

[8] B. Heigl, R. Koch, M. Pollefeys, and L. VanGool. “Plenoptic modeling and rendering from image sequences taken by a hand-held camera.” In *Proceedings of DAGM’99*, 1999.

[9] I. Ihm and S. Park. “Wavelet-based 3d compression scheme for interactive visualization of very large volume data.” *Computer Graphics Forum*, 18(1):3–15, March 1999.

[10] I. Ihm, S. Park, and R. K. Lee. “Rendering of spherical light fields.” In *Proceedings of Pacific Graphics’97*, pages 59–68. IEEE Computer Society Press, October 1999.

[11] A. Isaksen, L. McMillan, and S. J. Gortler. “Dynamically reparameterized light fields.” In K. Akeley, editor, *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 297–306. ACM SIGGRAPH, Addison Wesley, 2000.

[12] P. Lalonde and A. Fournier. “Interactive rendering of wavelet projected light fields.” In *Proceedings of Graphics Interface GI ’99*, pages 107–114, 1999.

[13] M. Levoy and P. Hanrahan. “Light field rendering.” In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 31–42. ACM SIGGRAPH, Addison Wesley, 1996.

[14] J. Li, H.-Y. Shum, and Y.-Q. Zhang. “On the compression of image based rendering scene: A Comparison among block, reference and wavelet coders.” *International Journal of Image and Graphics*, 1(1):45–61, January 2001.

[15] M. Magnor, A. Endmann, and B. Girod. “Progressive compression and rendering of light fields.” In *Proceedings of 5th International Fall Workshop Vision, Modeling and Visualization VMV 2000*, pages 199–203, November 2000.

[16] D. P. Mitchell. “Generating antialiased images at low sampling densities.” *Computer Graphics*, 21(4):65–72, 1987.

[17] I. Peter and W. Strasser. “The wavelet stream: Interactive multi resolution light field rendering.” In S. J. Gortler, editor, *Proceedings of the 12th Eurographics Workshop on Rendering*, pages 262–273, June 2001.

[18] H. Schirmacher, W. Heidrich, and H. P. Seidel. “Adaptive acquisition of lumigraphs from synthetic scenes.” In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Proceedings of EUROGRAPHICS’99)*, Vol. 18, No. 3, pages 151–159. Eurographics, Blackwell Publishers, 1999.

[19] E. J. Stollnitz, T. D. DeRose, and D. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann Publishers, Inc., 1996.

[20] Y. Wu, L. Luo, J. Li, and Y.-Q. Zhang. “Rendering of 3d wavelet compressed concentric mosaic scenery with progressive inverse wavelet synthesis (PIWS).” In *Proceedings of SPIE Visual Communications and Image Processing VCIP-2000*, pages 31–42, June 2000.

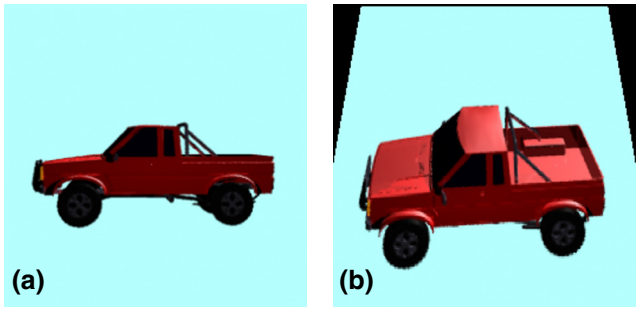

Figure 10: **Foreshortening: (a) horizontal view at *z = 0* and (b) perspective distortion due to an elevated camera position.**

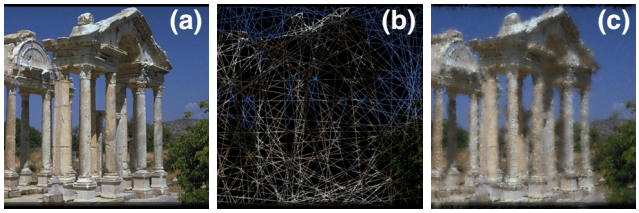

Figure 11: **2D example of the hierarchical scattered data interpolation scheme: (a) original, (b) scattered (75% unknown data) and (c) interpolated image using a gauss 5–tap filter.**

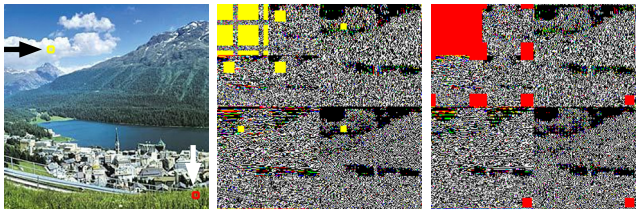

Figure 12: **Wavelet filter support for a local point update in 2D: original image (left) and transforms to level three, using a Daubechies–4 filter (middle) and a Daubechies–8 filter (right).**

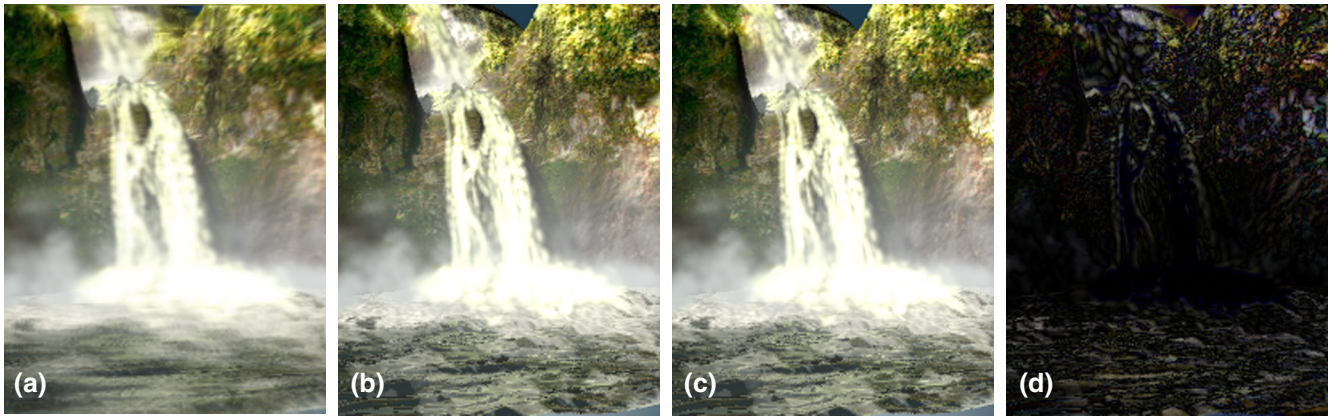

Figure 13: **Incremental update of the Falls light field using the local operators: (a) reconstructed view before the update, (b) input image, (c) reconstructed view after the update and (d) difference image used for the incremental update.**

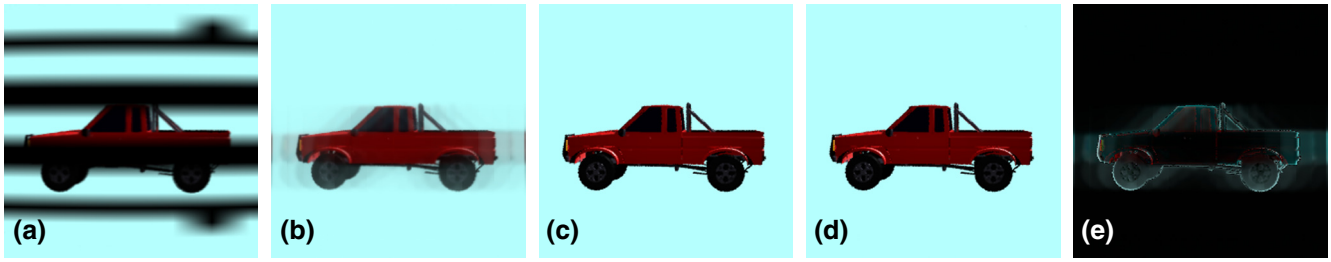

Figure 14: **The light field oracle modes demonstrated using the Truck light field: (a) view of the coarsely sampled light field containing undefined data, (b) same view after the transformation to the extended MRA, completing the first mode, (c) input image, (d) reconstructed view after the incremental update and (e) difference image between estimate and input image used for the incremental update in the oracle mode.**