IMAGE PROCESSING USING MULTI RATED GRID

Dr.NareshChetwani (Dean CSE/IT)[#1], Javed Akhtar Khan[#2]
1st Department of Computer Science & Engg.
1st NareshChetwani@takshshila.org
2nd Department of Information Technology
2nd javedaktarkhan@takshshila.org)
TAKSHSHILA INSTITUTE OF ENGINEERING & TECHNOLOGY, JABALPUR

Abstract: **We present a new data structure—the *multi rated grid*, that enables fast edge-aware image processing. By working in the multi rated grid, algorithms such as multi rated griding, edge-aware painting, and local histogram equalization become simple manipulations that are both local and independent. We parallelize our algorithms on modern GPUs to achieve real-time frame rates on high-definition video. We demonstrate our method on a variety of applications such as image editing, transfer of photographic look, and contrast enhancement of medical images.**

Keywords: **Computational Photography, Edge-Aware Image Processing, Multi rated Grid , Real-time Video Processing.**

1 Introduction

Nonlinear grid s have enabled novel image enhancement and manipulation techniques where image structure such as strong edges are taken into account. They are based on the observation that many meaningful image components and desirable image manipulations tend to be piecewise smooth rather than purely band-limited. For example, illumination is usually smooth except at shadow boundaries[1], and tone mapping suffers from haloing artifacts when a low-pass grid is used to drive local adjustment [2], a problem which can be solved with nonlinear grid s [3].In particular, the multi ratedgrid is a simple technique that smoothes an image except at strong edges [4]. It has been used in a variety of contexts to decompose an image into a piecewise-smooth large-scale base layer and a detail layer for tone mapping [5], and a wealth of other appli- A common drawback of nonlinear grid s is speed: a direct implementation of the multi rated grid can take several minutes for a one megapixel image. However, recent work has demonstrated acceleration and obtained good performance on CPUs, on the order of one second per megapixel [6]. However, these approaches still do not achieve real-time performance on high-definition content. In this work, we dramatically accelerate and generalize the multi rated grid , enabling a variety of edge-aware image processing applications in real-time on high-resolution inputs. Building upon the technique by Paris and Durand who use linear griding in a higher-dimensional space to achieve fast multi rated griding, we extend their high-dimensional approach and introduce the *multi ratedgrid*, a new compact data structure that enables a number of edge-aware manipulations. We parallelize these operations usingmodern graphics hardware to achieve real-time performance at HD resolutions. In particular, our GPU multi ratedgrid is two orders of magnitude faster than the equivalent CPU implementation.This paper makes the following contributions:

• The multi ratedgrid, a new data structure that naturally enables edge-aware manipulation of images.

• Real-time multi ratedgrid ing on the GPU, which is two ordersof magnitude faster than previous CPU techniques, enablingreal-time processing of HD content.

• Edge-aware algorithms. We introduce a number of realtime,edge-aware algorithms including edge-preserving painting,scattered data interpolation, and local histogram equalization.Our approach is implemented as a flexible library which is distributed in open source to facilitate research in this domain.The code is available at.We believe that the multi ratedgrid data structure is general and future research will develop many new applications.

2. Related Work

Multi rated Grid The multi rated grid is a nonlinear process thatsmoothes images while preserving their edges [7].

High-Dimensional Image Representation The interpretation of2D images as higher-dimensional structures has received much attention.Sochen et al. [1998] describe images as 2D manifolds embedded in a higher-dimensional space. For instance, a color image is embedded in a 5D space: 2D for *x* and *y* plus 3D for color. This enables an interpretation of PDE-based image processes in terms of geometric properties such as curvature. Our multi ratedgrid shares the use of higher-dimensional spaces with this approach. It is nonethelesssignificantly different since we consider values over the entire space and not only on the manifold. Furthermore, we store homogeneous values, which allows us to handle weighted functions and the normalization of linear grid s. Our approach is largely inspired by signal processing whereas Sochen et al. follow a differential geometry perspective.

In comparison, the multi ratedgrid does not encode the data into splines and allows direct access. This enables faster computation; in particular, it makes it easier to leverage the computational power ofmodern parallel architectures.

Multi ratedGrid

Multilevel methods have been developed for various application such as given below

- Elliptic PDEs
- Purely algebraic problems, with no physical grid; for example, network and geodetic survey problems.
- Image reconstruction and tomography (image processing)
- Optimization (e.g., the travelling salesman and long transportation problems)
- Statistical mechanics, Ising spin models.
- Quantum chromodynamics.
- Quadrature and generalized FFTs.
- Integral equations.

Now we are use this concept for Image processing .

A Simple Illustration

Before formally defining the multi ratedgrid, we first illustrate the concept with the simple example of an edge-aware brush. The

edge-aware brush is similar to brushes offered by traditional editing packages except that when the user clicks near an edge, the brush does not paint across the edge.When the user clicks on an image E at a location (xu, yu), the edgeawarebrush paints directly in the three-dimensional multi ratedgrid at position_

xu, yu,E(xu, yu)

The spatial coordinates are determinedby the click location, while the range coordinate is the intensity of the clicked pixel. The edge-aware brush has a smooth falloff in all three dimensions. The falloff along the two spatial dimensionsis the same as in classical 2D brushes, but the range falloff is specific to our brush and ensures that only a limited interval of intensity values is affected. To retrieve the painted value V in image space at location (x, y), we read the value of the multi ratedgrid at position

x, y,E(x, y)

We use the same terminology as Parisand Durand [2006] and call this operation *slicing*. In regions where the image E is nearly constant, the edge-aware brush behaves like a classical brush with a smooth spatial falloff. Since the range variations

are small, the range falloff has little influence. At edges, E is discontinuous and if only one side has been painted, the range

falloff ensures that the other side is unaffected, thereby creating adiscontinuity in the painted values V . Although the grid values are smooth, the output value map is piecewise-smooth and respects the strong edges of E because we slice ccording to E.

. *The image space result of the brush operation at a position x is obtained by interpolating the grid valuesat location.*

Definition

Data Structure The multi ratedgrid is a three dimensional array,where the first two dimensions (x, y) correspond to 2D position

in the image plane and form the spatial domain, while the thirddimension z corresponds to a reference range. Typically, the range axis is image intensity.

Sampling A multi ratedgrid is regularly sampled in each dimension. We name ss the sampling rate of the spatial axes and sr the

sampling rate of the range axis. Intuitively, ss controls the amount of smoothing, while sr controls the degree of edge preservation.

The number of grid cells is inversely proportional to the sampling rate: a smaller ss or sr yields a larger number of grid cells and requires more memory. In practice, most operations on the grid require only a coarse resolution, where the number of grid cells is much smaller than the number of image pixels. In our experiments, we use between 2048 and 3 million grid cells for the useful range of parameters. The required resolution depends on the operation and we will discuss it on a per-application basis.

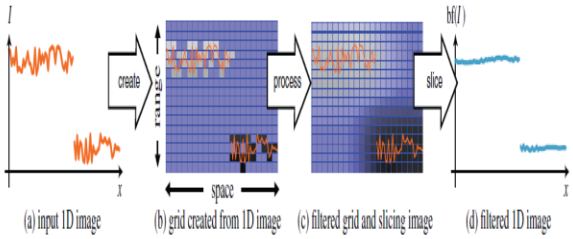
Data Type and Homogeneous Values The multi ratedgrid can store in its cells any type of data such as scalars and vectors. For many operations on the multi ratedgrid, it is important to keep trackof the number of pixels (or a weight w) that correspond to each grid cell. Hence, we store *homogeneous* quantities such as (wV,w) for a scalar V or (wR,wG,wB,w) if we deal

with RGB colors. This representation makes it easy to compute weighted averages:
$(w1V1, w1)+(w2V2, w2) = (w1V1 + w2V2), (w1 + w2)$
where
normalizing by the homogeneous coordinate $(w1 + w2)$ yields the expected result of averaging $V1$ and $V2$ weighted by $w1$ and $w2$. Intuitively, the homogeneous coordinate w encodes the importance of its associated data V . It is also similar to the homogeneous interpretation of premultiplied alpha colors [8].

Basic Usage of a Multi ratedGrid
Since the multi ratedgrid is inspired by the multi ratedgrid , we use it as the primary example of a grid operation. We describe a set of new manipulations in Section 4. In general, the multi ratedgrid is usedin three steps. First, we create a grid from an image or other user input. Then, we perform processing inside the grid. Finally, weslice the grid to reconstruct the output Construction and slicing are symmetric operations that convert between image and
grid space.

Grid Creation Given an image I normalized to [0, 1], s_s and s_r,the spatial and range sampling rates, we construct the multi ratedgrid
Γ as follows:
1. *Initialization*: For all grid nodes (i, j, k), $\Gamma(i, j, k) = (0, 0)$.
2. *Filling*: For each pixel at position (x, y):
$\Gamma([x/s_s] , [y/s_s] , [I(x, y)/s_r]) += (I(x, y), 1)$
where [·] is the closest-integer operator. We use the notation
$\Gamma = c (I)$ for this construction. Note that we accumulate both the image intensity and the number of pixels into each grid cell using
homogeneous coordinates.

Processing Any function f that takes a 3D function as input can be applied to a multi ratedgrid Γ to obtain a new multi ratedgrid
$\tilde{\Gamma} = f(\Gamma)$. For the multi ratedgrid , f is a convolution by a Gaussian kernel, where the variance along the domain and range dimensions are σ_s and σ_r respectively [8].



Extracting a 2D Map by Slicing
Slicing is the critical bilateralgrid operation that yields a piecewise-smooth output. Given a bilateral grid Γ and a reference image E, we extract a 2D value map M by accessing the grid at $x/s_s, y/s_s, E(x, y)/s_r$ using trilinear interpolation. We use the notation $M = sE(\Gamma)$. If the grid stores homogeneous values, we first interpolate the grid to retrieve the homogeneous vector; then, we divide the interpolated vector to access the actual data. Slicing is symmetric to the creation of the grid from an image.

3 GPU Parallelization
In developing the multi ratedgrid, one of our major goals was to facilitate parallelization on graphics hardware. Our benchmarks showed that on a CPU, the bottleneck lies in the slicing stage, where thecost is dominated by trilinear interpolation. We take advantage of hardware texture grid ing on the GPU to efficiently perform slicing. The GPU's fragment processors are also ideally suited to executing grid processing operations such as Gaussian blur.

Grid Creation
To create a multi ratedgrid from an image, we accumulate the valueof each input pixel into the appropriate grid voxel.


(a) input and stroke (b) intermediate (c) output

Grid creation is inherently a scatter operation since the grid position depends on a pixel's value. Since the vertex processor is the only unit that can perform a true scatter operation [Harris and Luebke 2004], we rasterizea vertex array of single pixel points and use a vertex shader to determine the output position. On modern hardware, the vertex processor can efficiently access texture memory. The vertex array consists of

(*x, y*) pixel positions; the vertex shader looks up the corresponding image value *I*(*x, y*) and computes the output position.

On older hardware, however, vertex texture fetch is a slow operation. Instead, we store the input image as vertex color attribute:each vertex is a record (*x, y, r, g, b*) and we can bypass the vertex texture fetch. The disadvantage of this approach is that during slicing, where the input image needs to be accessed as a texture, we must copy the data. For this, we use the pixel buffer object extension to do a fast copy within GPU memory.

Data Layout We store multi ratedgrids as 2D textures by tiling the zlevels across the texture plane. This layout lets us use hardware bilinear texture grid ing during the slicing stage and reduce the number of texture lookups from 8 to 2. To support homogeneous coordinates,we use four-component, 32-bit floating point textures. In this format, for typical values of *ss* = 16 and *sr* = 0.07 (15 intensitylevels), a multi ratedgrid requires about 1 megabyte of texture memory per megapixel. For the extreme case of *ss* = 2, the storage costis about 56 megabytes per megapixel. In general, a grid requires 16 × number of pixels/(*ss*2 × *sr*) bytes of texture memory. Grids that do not require homogeneous coordinates are stored as single component floating point textures and require 1/4 the memory. In our examples, we use between 50 kB and 40 MB.

Low-Pass Grid ing

As described in Section 2.3, the grid processing stage of the multi ratedgrid is a convolution by a 3D Gaussian kernel. In constructing the grid, we set the the sampling rates *ss* and *sr* to correspond to the bandwidths of the Gaussian σs and σr, which provides a good tradeoffbetween accuracy and storage [Paris and Durand 2006]. Since the Gaussian kernel is separable, we convolve the grid in each dimensionwith a 5-tap 1D kernel using a fragment shader.

Slicing

After processing the multi ratedgrid, we slice the grid using the input image *I* to extract the final 2D output. We slice on the GPU by rendering *I* as a textured quadrilateral and using a fragment shader tolook up the stored grid values in a texture. To perform trilinear interpolation, we enable bilinear texture grid ing, fetch the two nearestz levels, and interpolate between the results. By taking advantageof hardware bilinear interpolation, each output pixel requires only 2 indirect texture lookups.

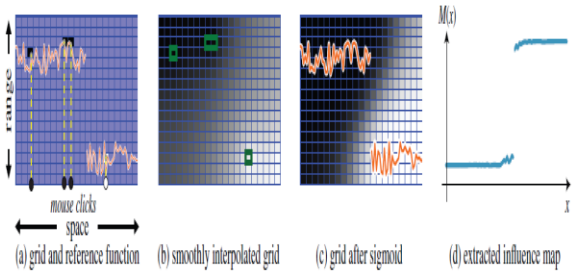
Further Acceleration

On current hardware, we can run multiple multi ratedgrid s per frameon 1080p HD video, but on older hardware, we are limited to a single grid per frame. For temporally coherent data, we proposean acceleration based on subsampling. A cell of the grid stores the weighted average of a large number of pixels and we can obtain a good estimate with only a subset of those pixels. For typical values of $\sigma s \in [10, 50]$ and $\sigma r \in [0.05, 0.4]$, using only 10% ofthe input pixels produces an output with no visual artifacts. We choose the 10% of pixels by rotating through a sequence of precomputed

Poisson-disk patterns to obtain a good coverage. To combat"swimming" artifacts introduced by time-varying sampling patterns, we apply a temporal exponential grid with a decay constant of 5 frames. This produces results visually indistinguishable from the full multi ratedgrid except at hard scene transitions.

Image Manipulation with the Multi rated Grid

The multi ratedgrid has a variety of applications beyond multi ratedgrid ing.The following sections introduce new ways of creating, processing and slicing a multi ratedgrid.



Conclusion

We have presented a new data structure, the multi rated grid,and use this concept for image processing that enables real-time edge-preserving image manipulation. By lifting image processing into a higher dimensional space, we are able to design algorithms that naturally respect strong edges in an image. Our approach maps well onto modern graphics hardware and enables real-time processing of high-definition video.

References

[1] Oh et al. 2001
[2] Chiu et al. 1993
[3] Tumblin and Turk 1999; Durand and Dorsey 2002
[4] Aurich and Weule 1995; Tomasi and Manduchi 1998; Smith and Brady 1997
[5] Durand and Dorsey 2002], flash/no-flash image fusion [Petschnigg et al. 2004; Eisemann and Durand 2004
[6] Durand and Dorsey 2002; Pham and van Vliet 2005; Paris and Durand 2006; Weiss 2006
[7] Aurich and Weule 1995; Smith and Brady 1997; Tomasi and Manduchi 1998
[8] Blinn 1996; Willis 2006

