Efficient Gradient Domain Compositing Using Quadtrees
Aseem Agarwala(SIGGRAPH 2007)

Presented by:
Yannis Atsonios,CS JHU

Outline

- Abstract
- Introduction
- Gradient-domain compositing
- Proposed Approach
- Experimental Setup/Results
- Proposed Future Work
- Conclusion

Abstract

- Description of a hierarchical approach to improve the efficiency of gradient domain compositing.
- Proposed method has complexity of $O(\sqrt{n})$
- Idea:Reduction of the problem space into a space in which much of the solution is smooth and then utilization of the smoothness pattern in an adaptive way by subdividing the problem domain using quadtrees.
- This approach will be tested in panoramic stitching and image region copy-paste.

Introduction

- Wealth of algorithms for combining regions of multiple photos or videos into a seamless composite operate in the gradient domain.
- Bottleneck of these approaches:Poor scalability.They try to solve a linear system and as the number of pixels is increasing(especially in multi-megapixel digital imagery) they run out of resources(time/space).
- Proposed method tries to deal with these problems.How?By solving reduced system …but to get visual identical results.
- How is this feasible? By observing that the difference between a simple color composite and its associated gradient domain composite is largely smooth…and this can be predicted a priori.
- The algorithm tries to solve this difference and in an adaptive way to subdivide the domain using Quadtree…with this way smoother areas of the solution are interpolated using fewer variables.

Introduction

- Poisson equation is well studied and a lot of algorithms are already proposed(such as multigrid methods due to Saad2003,other utilize GPU Bolz et al 2003, Szeliski 2006 introduced a preconditioner that boosts the convergence of the conjugate gradient solver).
- The disadvantage of these methods: Fundamentally :they do not address(attack) the inherent dimensionality of the problem directly,they solve a problem of bigger scale that in some cases is unnecessary. In terms of performance:They exhaust resources even if they have O(poly(n)),they run out of memory faster than time!
- A plethora of researchers tried adaptively to vary resolution when they solve the linear system or discretize the partial differential equation (a good paper is from Losasso et al 2004)
- The method that is proposed by Agarwala deviates from these methods in the manner that is applied in the gradient-domain compositing

Gradient-domain Compositing

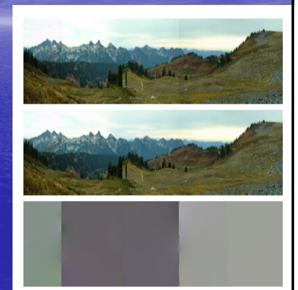
- Why to work on gradient domain?
- Gradient domain compositing hide seams between composited image regions that appear at the boundaries(high frequency artifacts) into low frequency variations that spread across the image.
- Gradient domain compositing is based on theory that human visual system is much more sensitive to local contrasts than to slow luminance and chrominance.
- Perez et al 2003 initiated the usefulness of working on this domain for a variety of image operations(mainly proposed method for seamless copy a region from one image into another).Followed by other researchers,namely Jia et al(2006) first optimizing the boundary of a copied region.
- Gradient domain compositing is crucial component in the state of the art techniques for seamless panoramic stitching after images have been aligned.Also extension for usage in video applications(Wang et al(2004))
- Revision:Computing large linear systems is not the only way to solve Poisson equation,other methods based on FFT(time:O(nlogn),space:O(n))

Gradient domain compositing(Mathematical Formulation)

- The method is performed for a single color channel of an image by re-ordering the image pixels into a vector x and solving the x that best matches the desired horizontal and vertical gradients ∇I_x, ∇I_y.
- Wealth of strategies for choosing these gradients.a)Compositing a region from image *Ia* into image *Ib* [Perez et al. 2003; Jia et al. 2006], the gradients inside this region are ∇I_a and the colors at the boundaries are fixed from image *Ib*.b)For the case of compositing multiple regions from multiple images, [Agarwala et al. 2004] is to use the gradient of the source image between any two pixels inside of one region, and the average of the gradients of the two source images between any two pixels that straddle a boundary between two regions.
- Formulation in matrix form Ax=b,x is length of n(one element for each pixel),A has most two non-zero elements per row.This system is overconstrained(ill-posed)…we search for the x that minimizes ||Ax-b||^2.Then we reduce the problem to the normal equations A'Ax=A'b,which is sparse and the square matrix A'A has at most five non-zero elements per row,hence we use an iterative solver.The initial condition is set as the image that would result by just copying colors (rather than gradients) from the source images.

Gradient domain compositing(Scalability)

- Unfortunately we have to solve n-element linear system and this is painful if n is large.
- If this approach used to composite a panorama…then we may run out of memory.
- Memory consumption for typical 50 megapixel panorama would require even 1 GB of memory just for computing one channel.
- Even the algorithm is polynomial(conjugate gradient solver),several instances of large input can lead to painfully long time to converge.
- The method proposed here tries to surpass this problem



Approach

- The approach is trying to solve the problem in a reduced space and makes the simple assumption that certain regions of the solution are smooth.
- Intuition:The initial residual $b-Ax_0$ will be zero for any pixel not adjacent to a seam ,since the colors of that pixel and its neighbors were just copied from one image and thus already satisfy the gradient constraints.
- $x=x_0+x_d$ where x_d is the difference between initial condition and final solution.Then $A'Ax_d = A'(b-Ax_0)$
- We observe that x_d will be very smooth away from the seams between the image regions.even if the final image x is not smooth anywhere.
- Making this observation ,we can represent each pixel in the smooth area not with one variable because is wasteful,and can be accurately interpolated with fewer variables with larger regions of support.
- Mechanism:High resolution used near seams and adaptively lower resolution in areas away from seams.
- How is achieved that?

Approach

- We substitute x=Sy,where y is a vector of dimension m<<n and S matrix m*n that is responsible for the transformation from reduced to full space.
- Near the seams every pixel will be represented by a variable(as in the full problem) and as we move way from the seams ,pixels will be in the smoother area and then can be interpolated as a weighted sum of several elements of y.
- What happens with the equations?
- $S'A'ASy_d=S'A'(b-ASy_0)$,left side and right side of the equations are pre-computed…the iterative solver has to address a m*m system than n*n sparse matrix vector multiplication.
- The hard part and the main idea of the approach:Define matrix S
- S is defined by an adaptive procedure.
- Adaptively we subdivide the problem domain using quadtree that is maximally subdivided to pixel-sized nodes along the seams.

Approach

- The quadtree is a pointer based tree which every non-leaf node has four children that subdivide space in 4 quadrants
- To ensure that we have gradual reduction in resolution away from these seams,the quadtree is restricted=>no two nodes that share an edge may differ in tree depth by more than one.
- With the quadtree constructed and the values of the vector y_d then the interpolation can be done by a simple traversal.
- S is encoding a bi-linear interpolation(can this be extended?)from quadtree nodes to pixels.How?Every x_d is computed by the values y_d at the four corners of the enclosed quadtree node

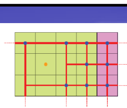
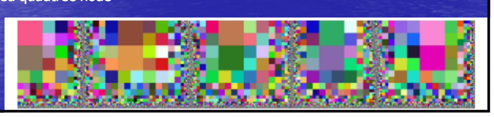


Figure 2 An inset of a grid of pixels in a larger gradient-domain composite. The green pixels are assigned to one source image, and the pink pixels to another; thus, a seam exists between them. The red lines are the boundaries of quadtree leaf nodes. The corners of quadtree nodes lie at pixel centers, and the quadtree is maximally subdivided along the seam. In the full linear system, each pixel is represented by one variable in the solution vector *x*. In the reduced linear system, variables in the solution vector *y* exist at the corners of the quadtree nodes (the blue dots), except along T-junctions where quadtree nodes of different sizes abut. We can interpolate the full solution *x* from a reduced solution *y* by computing $x = Sy$. This interpolation (which can be computed procedurally without ever building matrix *S*) is defined by the structure of the quadtree. Pixels that enclose a blue dot do not need to be interpolated; these values in *x* are just copied from the appropriate variable in *y* (which corresponds to a row in matrix *S* that is all 0 except for a single 1). Other pixels are interpolated; for example, the pixel enclosing the orange dot can be bi-linearly interpolated from the four corners of the enclosing quadtree node (which corresponds to a row of *S* with four non-zero values that sum to 1). One complication is that the lower-left corner of this node is not a variable in *y*, since it lies on a T-junction; the value here must first be linearly interpolated from the values of *y* at the blue dots above and below (not pictured).



Approach(Implementation)

- The algorithm has 3 major steps
- First:Construction of the quadtree
- Second:Solving the reduced scale linear system.Optimization is also done in the way of storing the matrices.Allocation only of O(m)memory
- Third:The interpolated solution x_d added to the initial x_0.We must add that we avoid saving the entire x_d ,and the interpolation stage can be done independently for sub-regions of the output image.
- Scale of the reduced space:How small is m in terms of n?
- m is linearly proportional to the number of leaf nodes in the quadtree,m=O(p).
- Observed that p= O(√n)(there is formal proof by Samet),growth of p is associated with how seams are chosen.

Experimental Setup/Results

- The algorithm compared against other two algorithms(a)hierarchical basis preconditioning HB,b)locally adapted hierarchical basis preconditioning LHB)
- LHB is one of the fastest algorithms for gradient domain problems.HB is somewhat older but doesn't need additional memory for preconditioning.
- The results are visually identical!

			Error		Time (s)			Memory (MB)		
Dataset	Mpixels	Vars (%)	RMS	Max	QT	HB	LHB	QT	HB	LHB
Plane	2.4	0.94	0.0108	0.36	3	371	26	13	96	227
St. Emilion	9.7	0.62	0.0132	1.37	9	3639	160	24	362	1044
Beynac	11.6	0.38	0.0103	1.25	8	3357	177	16	435	1252
Rainier	16.6	0.45	0.0157	1.15	14	6446	268	27	620	1790

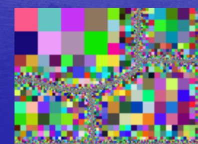
Table 1 Performance of three algorithms for several gradient-domain compositing problems. For each dataset, we show the number of megapixels, the number of variables per color channel in the reduced linear system as a percentage of the total number of pixels, and the error between the solutions computed using the reduced and full linear systems (error is measured using the 8-bit red channel, with both an average per-pixel RMS error and the maximum error across all pixels). We show the time and memory performance of three algorithms: quadtree based (QT), hierarchical basis preconditioning (HB), and locally adapted hierarchical basis preconditioning (LHB). Each panorama was stitched from five source images.

Dataset	Mpixels	Vars (%)	# sources	Time (s)	Memory (MB)
Sedona	34.6	0.47	6	29	52
Edinburgh	39.7	1.15	25	122	123
Crag	62.7	0.47	7	78	96
RedRock	83.7	0.46	9	118	112

Table 2 Performance of quadtree-based gradient-domain compositing for several very large panoramas.

Experimental Setup/Results(from http://agarwala.org/efficient_gdc)

- Image region copy-paste



Experimental Setup/Results(from http://agarwala.org/efficient_gdc)

- Panorama stitching

Future Work

- Extended this work for video,scalability concerns are even bigger.Idea using octrees.
- Shadow removal
- Removal of reflections in flash images.
- The algorithm probably will work in these domains because it creates an initial solution to the linear system whose residual is sparse,maybe this strategy of using direclty quadtrees in other gradient domain problems may collapse.

Conclusion

- Gradient domain Compositing is powerful method for compositing images and video regions,although it suffers from scalability issues and is expensive in terms of resources.
- The proposed method is an **approximation** that visually has identical results(!) and is extremely frugal in time and memory resources.

Questions


Thank you!