Abstract

- Description of a hierarchical approach to improve the efficiency of gradient domain compositing.
- Proposed method has complexity of $O(n\log n)$.
- 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

- 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 ($\text{time}: O(n\log n)$, space: $O(n)$)

Outline

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

Conclusion

- Gradient-domain Compositing
- 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 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.
Gradient domain compositing (Mathematical Formulation)

- The method is performed for a single color channel of an image by reordering the image pixels into a vector \( x \) and solving the \( x \) that best matches the desired horizontal and vertical gradients \((f_x, f_y)\).
- Weights of strategies for choosing these gradients: a) Compositing a region from image \( I_a \) into image \( I_b \) (Peraza et al., 2001; Jia et al., 2006), the gradients inside this region are \( f_x \) and the colors at the boundaries are fixed from image \( I_{a,b} \).
- For the case of compositing multiple regions from multiple images, \((\text{Agarwala et al., 2004})\) 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 \( Ax = b \) is length of \((x)\) (one element for each pixel). A has most two zero-nonzero elements per row. This system is overconstrained (9 per pixel), so we search for the \( x \) that minimizes \( |Ax-b|^2 \). Then we reduce the problem to the normal equations \( A^TAx=A^Tb \) which is spare and the square matrix \( A^TA \) 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 a \( n \)-element linear system and this is painful if \( n \) is large.
- If this approach used to compose a panorama, then we may run out of memory.
- Memory consumption for typical \( 50 \) megapixel panorama would require over \( 1 \) GB of memory just for computing one channel.
- Even the algorithm is (asymptotically) \( O(n^3) \) gradient solver, several instances of large input can lead to partially 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. Hence even if the corner is not violated and neighbors were just copied from one image and thus already satisfy the gradient constraints.
- \( x_{opt} = x_0 \) where \( x_0 \) is the difference between initial condition and final solution. Then \( A^TAx = A^Tb \).
- We observe that \( x_{opt} \) will be very smooth away from the seams between the image boundaries 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

- The quadtree is a pointer based tree which every non-leaf node has four children that make up space in 4 quadrants.
- To ensure that we have gradual reduction in resolution away from these seams, the quadtree is restricted to \( n \) to 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 \( x \) then interpolation can be done by a simple traversal.
- By introducing a \( b \)-linear interpolation can this be extended? from quadtree nodes to pixels how many \( x \) is computed by the values \( x_{opt} \) at the four corners of the enclosed quadtree node.
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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Descent</th>
<th>Tangent</th>
<th>Project</th>
<th>Hue</th>
<th>Scale</th>
<th>Smooth</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sow</td>
<td>1.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>Ender</td>
<td>0.6</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>Outdoor</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.9</td>
<td>0.7</td>
</tr>
</tbody>
</table>

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

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 directly octrees in other gradient domain problems may collapse.

Conclusion

- Gradient domain Compositing is a 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!