Interactive Large-scale Image Editing using Operator Reduction

Manhee Lee
Inha University

Won-Ki Jeong
UNIST

Hanspeter Pfister
Harvard University

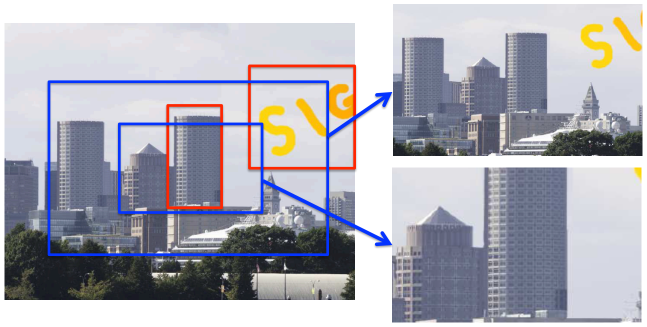

Figure 1: *Overview of the editing system. Red rectangles are editing operators. Blue rectangles are visible regions in different zoom levels and view points.*

With advent of advances in digital imaging technology, large-scale high-resolution images become commonplace these days. Automated digital cameras or scientific instruments can produce tens of millions pixel images at interactive rates, and modern photo editing software can stitch such high-resolution images to create much larger-scale panoramic images with minimal user efforts. Those large-scale images convey rich information of objects and scenes by providing both large field-of-view when zoomed out and extremely fine details when zoomed in, which is not available in normal-sized images. In addition, such high-resolution imaging is becoming an important research tool for scientific discoveries [Jeong et al. 2009].

In this poster, we introduce our on-going work on developing acceleration techniques for interactive editing of large images. Our method is inspired by the previous work [Jeong et al. 2011]. The major limitations of [Jeong et al. 2011] is that the computational complexity depends on the number of overlapping editing operators, and some editing operators, e.g., push-pull blending or stitching, still depend on the local multi-resolution data hierarchy for fast computation. To address these problems, we propose a novel operator reduction technique that can reduce multiple editing operators to a single operator, using the domain subdivision method with a GPU-friendly image editing pipeline.

Editing System Overview In our system, the input image, editing operators, and the current visible region are defined as axis-aligned rectangles, i.e., 2D bounding boxes (Fig 1). The size, location, and scale (i.e., level) of the editing operator is defined when it is created. If the user changes the view point, e.g., zooming or panning, then the location and the size of the viewing bounding box is changed accordingly (Fig 1 blue rectangles). The final image on the display is generated on-the-fly by applying the operators to the visible portion of the base image. Note that the visible portion of the image can be quickly retrieved from the image repository (tile-based, multi-resolution image data hierarchy), but the editing is done on-the-fly on a currently displayed screen-sized image.

Subdivision of the Image Domain A naive approach to apply editing operators to the current viewing region is finding all the intersected operator bounding boxes with the viewing bounding box and apply them in a temporal order (i.e., the order of the creation of operators). One drawback of this approach is that the computation time increases linearly as the number of overlapped operators increases. However, our proposed system can optimize the process by grouping multiple operators to a single operator, we call it *operator reduction*. The requirement for the operator reduction is that the location and size of the operators should be identical, which is not feasible in practice. To fulfill the requirement for the operator reduction, we propose a method that subdivides the image domain. The main idea of this approach is that instead of storing a bounding box for an editing operator explicitly, we assign the operator to a subset of the subdivision of the image domain. In this setup, the input domain is composed of a set of non-overlapping 2D bounding boxes where each box manages an operator list to keep track of editing operations applied to the corresponding region.

Reduction of Editing Operators As discussed in the previous section, multiple operators can be assigned to a single bounding box. For a certain class of operators, a set of successive operations can be combined and expressed as a single operation. A simple example is the paint brush. A paint brush operator blends the paint color with the background color using some weight factors. When two paint brush strokes are overlapped, we can convert two paint brush operators into a single paint brush by creating a new paint brush with the blended color. For complicated operators, we convert the operator to an approximated linear operator for speeding up. For example, the Poisson blending, we first compute the solution of Poisson equation solver, and then convert the blended image to the base (original) image and the detail image. That means, if the user changes zoom levels or view points, the Poisson blending operator does not need to re-calculate the solution but add the detail map to the currently visible image.

GPU Acceleration of Editing Operators In addition to the operation reduction, each editing operator can be further accelerated using the GPU. Most editing operators we implemented are GPU-friendly because pixel values can be calculated in parallel without accessing their neighbor pixels. Therefore, we implement each operator as a shader subroutine, and each bounding box is a GPU kernel that calls the corresponding shader code. To speed up the Poisson blending operator, we implement a GPU multi-grid Conjugate-Gradient Poisson equation solver. We used a compressed sparse row data structure to store sparse matrix, and sparse matrix-vector multiplication is done by using CUSPARSE and CUBLAS library.

Preliminary Results We tested our prototype editing system on an Windows PC equipped with an NVIDIA GeForce GTX 480 GPU. We are able to apply up to 50 editing operators at a rate of 30 frames per second on a 1920 x 1200 sized screen independent from the input image size.

References

Jeong, W.-K., Beyer, J., Hadwiger, M., Vazquez-Reina, A., Pfister, H., and Whitaker, R. T. 2009. Scalable and interactive segmentation and visualization of neural processes in EM datasets. *IEEE Trans. Vis. Comp. Graph. 15*, 6.

Jeong, W.-K., Johnson, M. K., Yu, I., Kautz, J., Pfister, H., and Paris, S. 2011. Display-aware image editing. In *International Conference on Computational Photography*, 1–8.

Copyright is held by the author / owner(s).
SIGGRAPH Asia 2011, Hong Kong, China, December 12 – 15, 2011.
ISBN 978-1-4503-0807-6/11/0012