Our Work-Efficient GPU Algorithm

Our GPU algorithm is the first and only GPU level set segmentation algorithm that is both work-efficient and step-efficient. Our GPU algorithm executes (O(n)) steps where n is the size of the active computational domain. For parallel stream compaction we use the work-efficient and step-efficient algorithm described by Harris et al. 2007 and Sengupta et al. 2006, and included in the open-source CUDPP library 2010.

Initialization

- Initialize the level set field (green) in parallel. Each thread initializes one level set field element.
- In a separate computational kernel, examine the level set field for active coordinates (blue). Write all such coordinates to a scratchpad using the coordinates themselves as array indices. Each thread examines one level set field element.
- Compact the scratchpad in parallel to produce a dense list of active coordinates.

Update

- Update the level set field in parallel. Each thread updates the level set field at one active coordinate.
- Output new active coordinates in parallel. Each thread examines the neighborhood around one old active coordinate (grey) and outputs any new active coordinates (blue) to a series of auxiliary buffers. The choice of auxiliary buffer is determined by the position of the old active coordinate relative to the new active coordinate. Each isolated auxiliary buffer is free of duplicates, but taken collectively the auxiliary buffers may contain duplicates.
- Remove duplicates from the auxiliary buffers in parallel without atomic memory operations or sorting:
  - In parallel, tag a scratchpad at all the coordinates in the first auxiliary buffer.
  - Examine the next auxiliary buffer. In parallel, remove all previously tagged coordinates (red) from the auxiliary buffer. Tag the scratchpad all the coordinates that have not been previously tagged (blue).
  - Repeat this process for each of the remaining auxiliary buffers. This is correct and free of race conditions since each step examines one auxiliary buffer and there are no duplicates within each auxiliary buffer.

Our Algorithm for Exploiting Temporal and Spatial Coherence

In contrast to previous narrow band algorithms, we make the observation that even computations near the level set surface can be avoided in regions where the level set field has already converged. This observation motivates our novel algorithm for tracking the active computational domain according to both the temporal and spatial derivatives of the level set field.

Our algorithm for tracking the active computational domain. Image data is shown in grey; currently segmented elements are shown in green; and intermediate results for computing the active computational domain are shown in blue. The active computational domain is outlined in black, and inactive elements are shown as partially transparent. The user places a seed to initialize the level set field and the initial active computational domain is determined according to the spatial derivatives of the level set field (a). During each iteration the level set field is updated at all active elements (b). The new active computational domain is computed according to the temporal and spatial derivatives of the level set field (c). If the new active computational domain is empty (d) then our segmentation has globally converged (e). Otherwise we go to (b).