Eurographics Symposium on Parallel Graphics and Visualization (2025)
G. Reina, S. Rizzi, C. Gueunet (Editors)

A Transparent and Efficient Extension of IceT for Parallel Compositing on Non-Convex Volume Domain Decompositions: Supplementary Document

Paul Hempel*[1], Aryaman Gupta*[1], Ivo F. Sbalzarini[1,2,3] and Stefan Gumhold[1]

[1]Dresden University of Technology, Faculty of Computer Science, Dresden, Germany
[2]Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany
[3]Center for Systems Biology Dresden (CSBD), Dresden, Germany
* The authors contributed equally.

```
/* Copy fragments until an inactive one is encountered.
 */
#define CT_WRITE_PIXEL(dest)                                                \
{                                                                           \
    /* Count active fragments. */                                           \
    IceTLayerCount pixel_size = 0;                                          \
                                                                            \
    /* Leave room to store the number of active fragments and               \
     * remember the location. */                                           \
    IceTLayerCount *const pixel_size_out = (IceTLayerCount *)dest;          \
    dest += sizeof(IceTLayerCount);                                         \
                                                                            \
    /* Copy active fragments, which must come first. */                     \
    for (IceTLayerCount layer = 0; layer < _num_layers; ++layer) {          \
        if (_color[layer*CTL_COLOR_CHANNELS + CTL_ALPHA_CHANNEL] == 0) break;\
        CTL_WRITE_FRAGMENT(layer, dest);                                    \
        ++pixel_size;                                                       \
    }                                                                       \
                                                                            \
    /* Write the number of active fragments to the reserved location. */    \
    *pixel_size_out = pixel_size;                                           \
    CT_ACTIVE_FRAGS += pixel_size;                                          \
}
```

Figure S1: *A custom macro defined to encode a single active pixel in a layered image, invoked by the* `compress_template_body.h` *template for active pixel encoding. The macro writes the active fragments within the pixel to the encoded buffer, counts them, and writes the count to the preceding location in a single pass through the fragment values.*

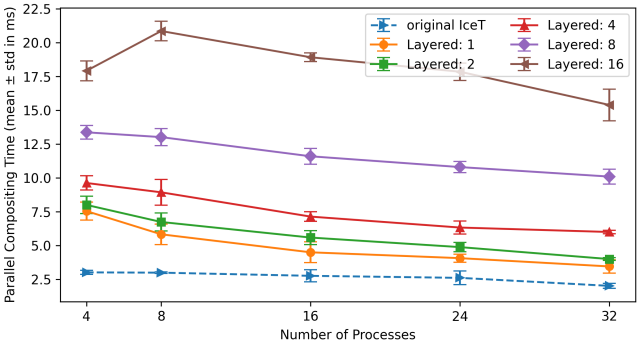


Figure S2: *Parallel compositing times (mean ± standard deviation over a 360° orbit) over non-convex decompositions of the Richtmyer-Meshkov dataset with 1 non-adjacent block per* ***PE*** *(Layered: 1) up until 16 non-adjacent blocks per* ***PE*** *(Layered: 16), compared with the original, non-layered IceT as a baseline (dashed).*

1. Active pixel encoding - Code macro

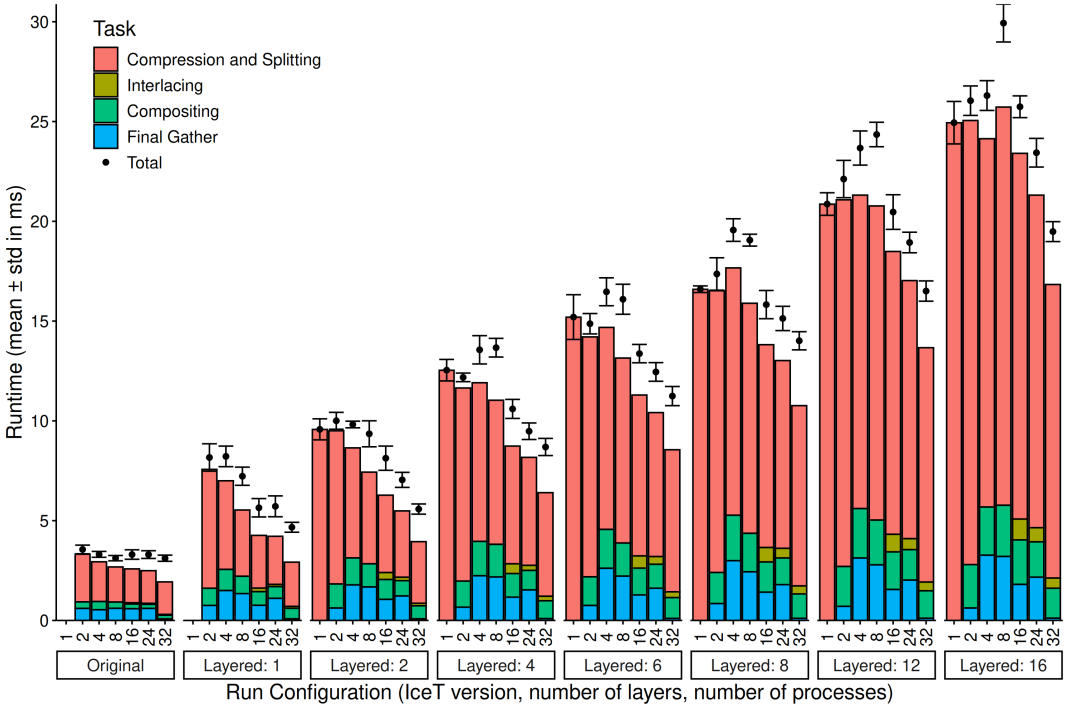
Section 5.1 in the main document described a custom macro responsible for encoding a single active pixel in a layered image. Fig. S1 shows this macro. As described in the main document, the macro finds inactive pixels by checking whether the first fragment in the pixel is empty. For active pixels, it counts the number of active fragments within the pixel and writes this value to the encoded output.

2. Benchmarks and Results

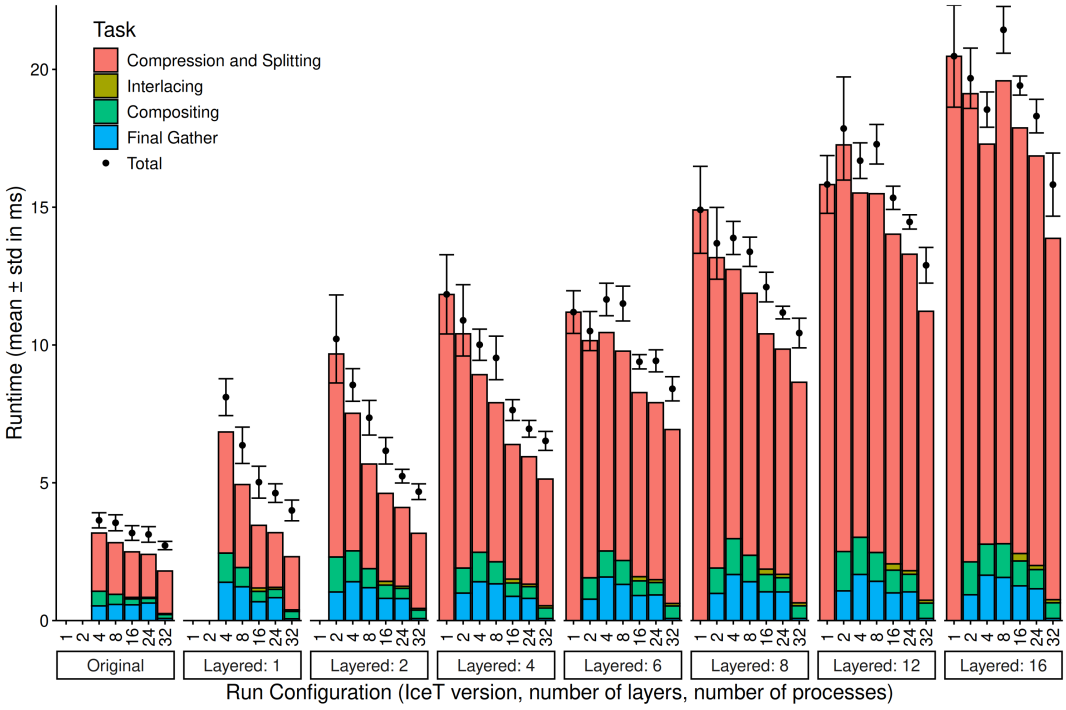
As described in Section 7 of the main document, we performed micro-benchmarks of our implementation using the Rayleigh-Taylor and Richtmyer-Meshkov datasets. Fig. S2 presents the results for the Richtmyer-Meshkov dataset, analogous to Fig. 3 of the main document. Parallel compositing times are generally shorter for this dataset because the rendering contains more empty pixels. General scaling trends are the same for both datasets.

Fig. S3 reports fine-grained performance profiling results on the Rayleigh-Taylor and Richtmyer-Meshkov datasets. For performance analysis, IceT times several steps in the parallel compositing process. These measurements cannot be disabled, so profiling data is available for every run without additional overhead. The timings shown in Fig. S3a correspond to the same runs that produced Fig. 3 in the main document, and the timings in Fig. S3b correspond to Fig. S2. The parallel compositing strategy used for both is `radix-k` with $k = 8$.

Most of the runtime is taken up by the initial compression of the input image using active pixel encoding. Since each process only has to compress its own starting partition, compression time decreases as more **PE**s are added: More processes means more partitions with fewer pixels each, and thus less work per process. Com-



(a) *Profiling results for the Rayleigh-Taylor dataset.*



(b) *Profiling results for the Richtmyer-Meshkov dataset.*

Figure S3: *Timings from IceT's built-in profiling, showing how compositing time is divided between sub-tasks. The original version of IceT (original) composited regular images, our modified version used layered images with layer counts between one (Layered: 1) and 16 (Layered: 16). Measurements are averaged over 90 repetitions of a 360° orbit in five steps of 72° around the dataset. Stacked bars represent sub-tasks timed by IceT, black dots the overall time to composite one frame. Error bars show standard deviation.*

pression time also includes the time of splitting the local image before each round of radix-k compositing. Since there is one round for every multiple of the group size, the decrease in compression time is somewhat offset by an increase in splitting time if more than eight processes are used. If there are multiple rounds, i.e. there are more than eight **PE**s, IceT also interlaces the image by shuffling partitions once before the first round.

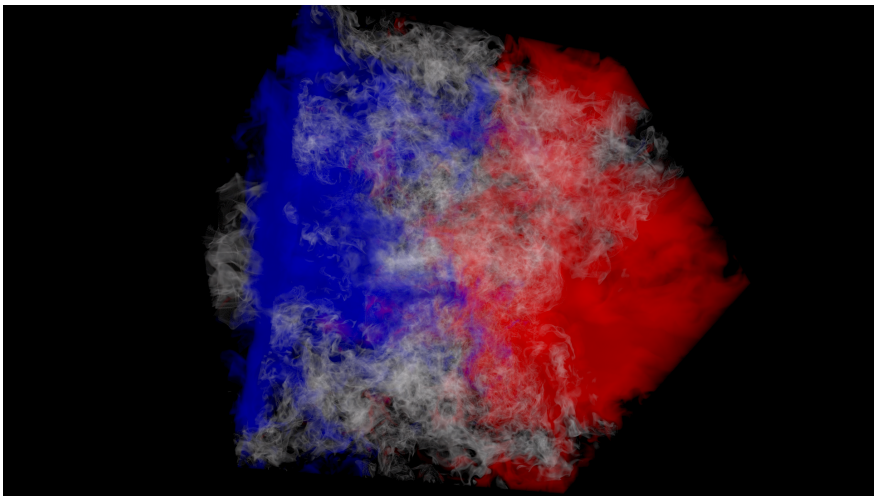
The 'compositing' sub-task reports the runtime for the merging sub-images after they are distributed using MPI. For the original version of IceT, this means blending the colors at each pixel. For layered images, the per-pixel fragment lists from both source images are merged into one, ordered by depth (see Section 5.4 in the main document). Compositing times decrease with increasing numbers of **PE**s despite the fact that each process needs to composite more images.

Once all parallel compositing rounds are complete, image partitions at each process are gathered at the root process to form the complete image. Since the data is sent uncompressed, this 'final gather' takes a non-negligible amount of time, even though it happens only once. It also takes longer for layered images, because depth is gathered as well, doubling the amount of data.

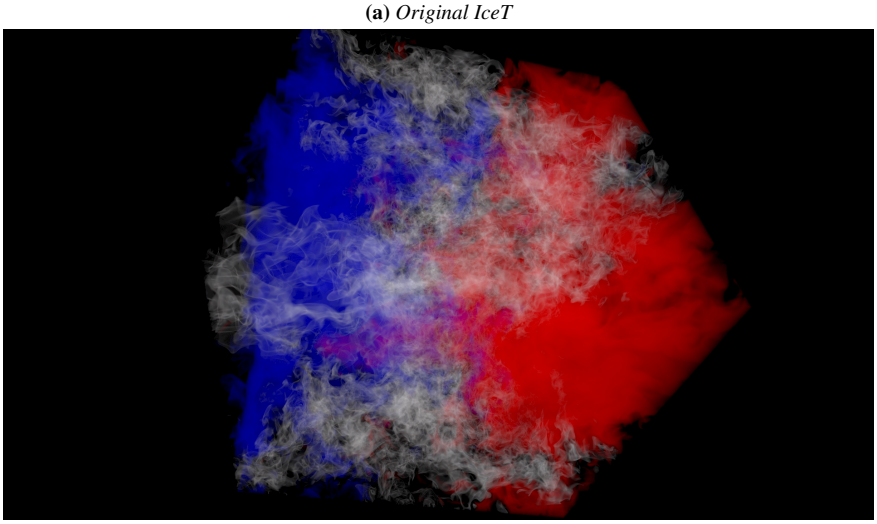
All remaining time is spent on MPI communication to distribute sub-images between processes. Note, however, that this MPI communication runs asynchronously with compositing, so the remaining time does not represent the total time required to distribute images using MPI.

3. Visual comparisons

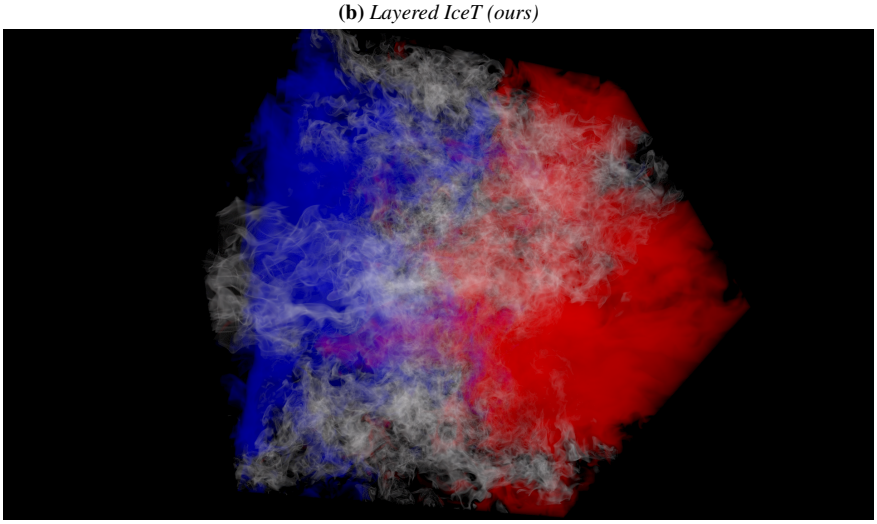
Figs. S4 and S5 show additional visual comparisons on the Rayleigh-Taylor and Richtmyer-Meshkov dataset respectively, supplementing the images shown in Fig. 4 of the main text. See the other supplementary file for full-resolution images on both datasets generated using various run configurations.



(a) *Original IceT*

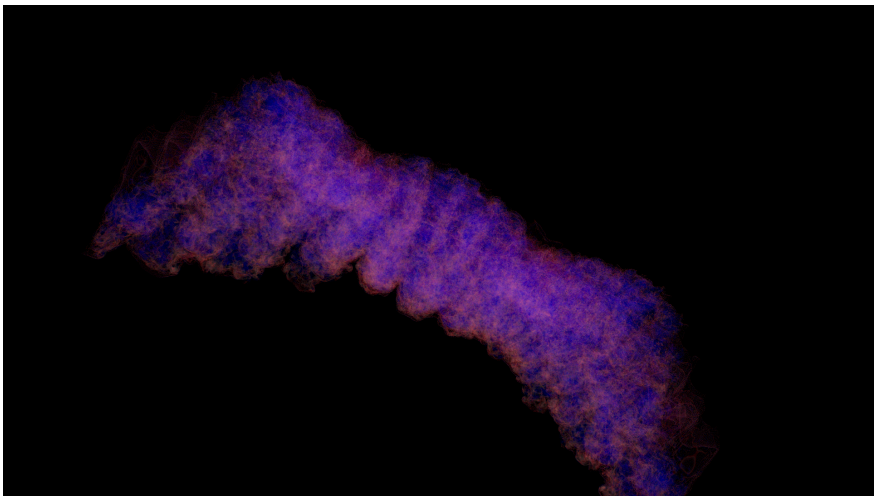


(b) *Layered IceT (ours)*

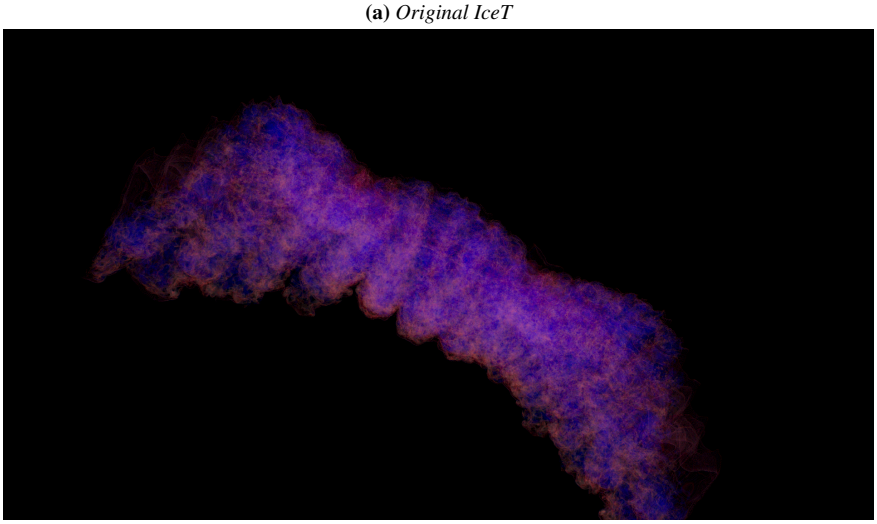


(c) *Reference*

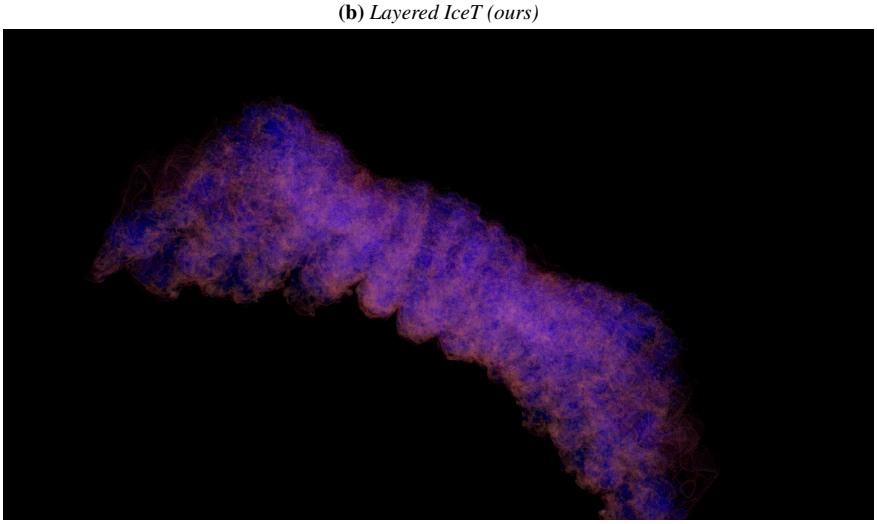
Figure S4: *Visual comparison of compositing output by the original IceT (a) and Layered IceT (b) on a non-convex decomposition of the Rayleigh-Taylor dataset over 32* ***PE****s, and a reference rendering for the dataset (c).*



(a) *Original IceT*



(b) *Layered IceT (ours)*



(c) *Reference*

Figure S5: *Visual comparison of compositing output by the original IceT (a) and Layered IceT (b) on a non-convex decomposition of the Richtmyer-Meshkov dataset over 32 **PE**s, and a reference rendering for the dataset (c).*