Temporal interpolation of 4D PC-MRI blood-flow measurements using bidirectional physics-based fluid simulation

Additional materials

Algorithm 1: Find valid simulation cells.

```
1  Input: user_selected_planes – the planes selected by the user
2  Output: Valid simulation cells
3  Initialize all cells, that are known to be of type SOLID, SOURCE or SINK;
4  Mark all other cells to be of type UNDEF;
5  queue ← ∅;
6  foreach x ∈ user_selected_planes do
7  |  T ← x.type; //the type of the selected plane is stored in T
8  |  queue ← ∅;
9  |  foreach cell c covered by x do
10 |  |  foreach neighbour cell n of c do
11 |  |  |  updateCell(n,queue,T);
12 |  |  end
13 |  end
14 |  while queue ≠ ∅ do
15 |  |  c ← queue.popfirst;
16 |  |  foreach neighbour cell n of c do
17 |  |  |  updateCell(n,queue,T);
18 |  |  end
19 |  end
20 |  //all cells are now sweeped
21 end
22 Make all source and sink cell neighbouring at least one VALID cell also
   VALID
```

Algorithm 2: Update cell

```
1  Input: A cell c, the current queue, the type of the selected plane T
2  Output: Updated queue
3  if c.type ∈ {VALID, SOLID, SOURCE, SINK} then
4  |  return;//we do not need to update this cell
5  end
6  if c.type is UNDEF then
7  |  if T is SOURCE then
8  |  |  Mark c to be of type SOURCE_REACHABLE;
9  |  end
10 |  if T is SINK then
11 |  |  Mark c to be of type SINK_REACHABLE;
12 |  end
13 |  queue ← queue ∪ {c}; //enqueue this cell
14 end
15 if c.type is SOURCE_REACHABLE and T is SINK then
16 |  Mark c to be of type VALID;
17 |  queue ← queue ∪ {c}; //enqueue this cell
18 end
19 if c.type is SINK_REACHABLE and T is SOURCE then
20 |  Mark c to be of type VALID;
21 |  queue ← queue ∪ {c}; //enqueue this cell
22 end
```