Supplemental Material

The supplemental material is organized as follows:

In Section A we give pseudocode of the algorithms described in Section 3, and show additional plots B.

A. Algorithms

```
in   : dim                                          // image dimensions
inout: disMap                                       // disocclusion map

// Spread disocclusions
numLevels ← ⌈log_2(max(dim.x, dim.y))⌉
level ← 1
m ← 1
srcMap ← disMap
dstMap ← empty
while level ≤ numLevels do
    // Relevant extents and offsets for horizontal and vertical neighbors
    hvNeighbors ← {(r,(−m,0)), (l,(m,0)), (t,(0,−m)), (b,(0,m))}
    // Relevant extents and offsets for diagonal neighbors
    diagNeighbors ←
      {(rt,(−m,−m)), (lt,(m,−m)), (rb,(−m,m)), (lb,(m,m))}
    foreach p = (x,y) ∈ {0,...,dim.x} × {0,...,dim.y} do
        spread ← srcMap(p)
        // Spread disocclusion from horizontal and vertical neighbors
        foreach (e,o) ∈ hvNeighbors do
            spread.e ← max(spread.e, srcMap(p+o).e − m)
        end
        // Spread disocclusion from diagonal neighbors
        foreach (e,o) ∈ diagNeighbors do
            if all(srcMap(p+o).e − (m,m) > (0,0)) then
                spread.e ← max(spread.e, srcMap(p+o).e − (m,m))
        end
        dstMap(p) ← spread
    end
    level ← level + 1
    m ← m · 2
    swap(srcMap, dstMap)
end
disMap ← srcMap
```

Algorithm 2: Pseudocode for spreading of disocclusions caused by motion. Input is a disocclusion map which is initialized with Algorithm 1.

```
in  : dim                                           // image dimensions
in  : dMap                                          // first layer depth map
in  : vMap                                          // first layer velocity map
out: disMap                                         // disocclusion map

(vDxMap, vDyMap) ← forward_differences(vMap)
// Initialize the disocclusion map
foreach (x,y) ∈ {0,...,dim.x} × {0,...,dim.y} do
    currentD ← dMap(x,y)
    rightD ← dMap(x+1,y)
    topD ← dMap(x,y+1)
    vDx ← vDxMap(x,y)
    vDy ← vDyMap(x,y)
    // Compute disocclusion extents at vertical edges
    disocclusionX.(l,r,t,b) ← (0,0,0,0)
    if vDx.x > 0 then
        if currentD > rightD then
            disocclusionX.r ← vDx.x
            if vDx.y > 0 then
                disocclusionX.t ← vDx.y
            else
                disocclusionX.b ← −vDx.y
        else
            disocclusionX.l ← vDx.x
            if vDx.y < 0 then
                disocclusionX.t ← −vDx.y
            else
                disocclusionX.b ← vDx.y
        end
    end
    // Compute disocclusion extents at horizontal edges
    disocclusionY.(l,r,t,b) ← (0,0,0,0)
    if vDy.y > 0 then
        if currentD > topD then
            disocclusionY.t ← vDy.y
            if vDy.x > 0 then
                disocclusionY.r ← vDy.x
            else
                disocclusionY.l ← −vDy.x
        else
            disocclusionY.b ← vDy.y
            if vDy.x < 0 then
                disocclusionY.r ← −vDy.x
            else
                disocclusionY.l ← vDy.x
        end
    end
    disMap(x,y) ← max(disocclusionX, disocclusionY)
end
```

Algorithm 1: Pseudocode for initialization of the disocclusion map for disocclusions caused by motion.

B. Additional graphs and figures

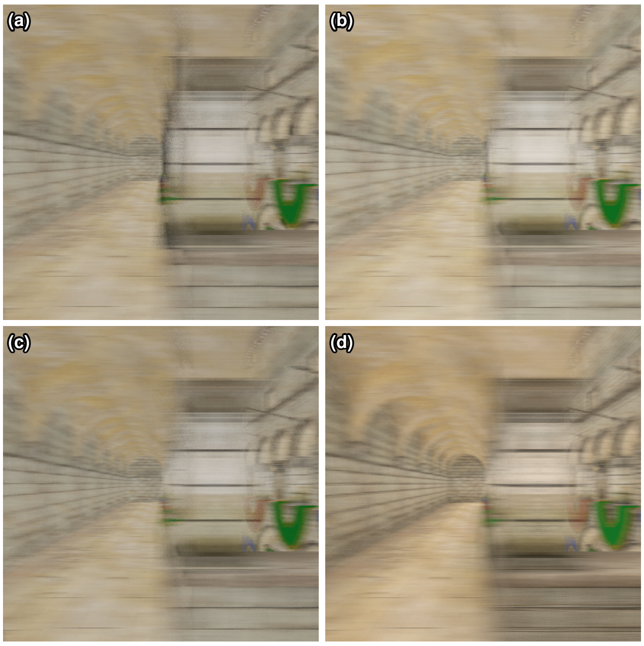

Figure 15: *Comparison of using a two depth layer (**a**) without minimum z-separation, two depth layers with correct z-separation (**b**) and infinite depth (**c**) against Blender reference (**d**) for multiple disocclusions. While there are still some artifact remaining when using the second layer, they are hardly noticeable during animation.*

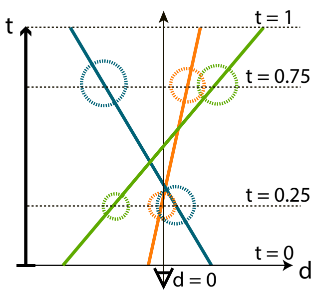

Figure 16: *The t-fragments for $t = 0.25$ (see Figure 6 in the paper) visualized according to their world-space distance from the viewing ray over time.*

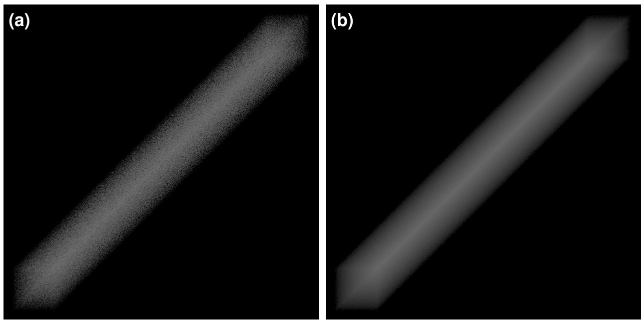

Figure 17: *Comparison of our approach (**a**) against Blender reference (**b**) for very large motion vectors.*

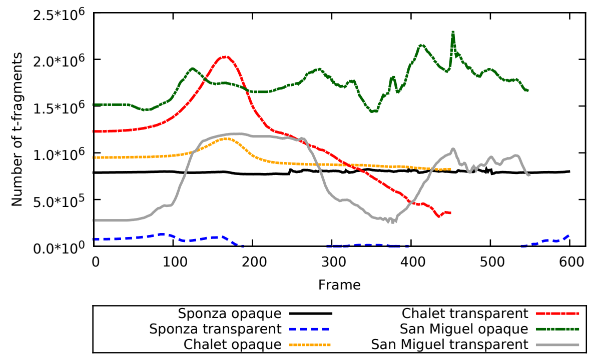

Figure 18: *Number of opaque and transparent fragments generated using the disocclusion map for early fragment culling.*