APPENDIX A
EXTENDED EXPLANATION OF ALGORITHMS

This part further explains the global EDT integration algorithm. In Algorithm 4, the wavefront propagates the distance values in $frontierB$ in GPU hash table $GHash$. For all neighbors whose distance value can be reduced by cur, we separate them into two streams: For those outside the local volume, we add nbr to $frontierB$ for the next-round propagation; otherwise, we add nbr to $frontierC$. With $frontierC$, Algorithm 5 propagates the global observation from the GPU hash table $GHash$ to the local EDT E. This batch of memory within the local volume is then copied to CPU and broadcast by ROS. On the other hand, the updated portion of the global EDT in the GPU hash table is streamed to the CPU hash table, which adds additional overhead.

Algorithm 4: PARWAVEFRONT (lower outside)

Input: Initial frontiers $frontierB$, $frontiersC$, current local EDT E, and global EDT in GPU hash table $GHash$
Output: Updated local EDT and global EDT

while $frontierB.size \neq 0$ **do**
  CHOOSELEVEL($frontierB$)
  forall $cur \in frontierB$ **do**
    $frontierB \leftarrow frontierB \backslash \{cur\}$
    forall $nbr \in cur.nbrs$ **do**
      if $nbr \notin E.volume$ **then**
        if DIST$(GHash, nbr) > ||cur.parent - nbr||_2$ **then**
          DIST$(GHash, nbr) \leftarrow ||cur.parent - nbr||_2$
          $nbr.parent \leftarrow cur.parent$
          $frontierB \leftarrow frontierB \cup \{nbr\}$
      else if DIST$(E, nbr) > ||cur.parent - nbr||_2$ **then**
        $frontierC \leftarrow frontierC \cup \{cur\}$
return E, $GHash$

Algorithm 5: PARWAVEFRONT (lower inside)

Input: Initial frontier $frontierC$, current local EDT E, and global EDT in GPU hash table $GHash$
Output: Updated local EDT and global EDT

while $frontierC.size \neq 0$ **do**
  CHOOSELEVEL($frontierC$)
  forall $cur \in frontierC$ **do**
    $frontierC \leftarrow frontierC \backslash \{cur\}$
    forall $nbr \in \{x | x \in cur.nbrs \wedge x \in E.volume\}$ **do**
      if DIST$(E, nbr) > ||cur.parent - nbr||_2$ **then**
        DIST$(E, nbr) \leftarrow ||cur.parent - nbr||_2$
        $nbr.parent \leftarrow cur.parent$
        $frontierC \leftarrow frontierC \cup \{nbr\}$
return E, $GHash$

Note that the pseudo-code is written in a sequential fashion for ease of understanding. In actual software, details such as array allocation should be taken care of. At the end of each round of propagation, the new frontier is reconstructed by the parallel allocation algorithm. Given several queues $localQ$ on different warps of a thread-block, the algorithm first counts the index of each element using prefix-sum. After that, multiple threads copy elements from the $localQ$ to the frontier for passing in the next-round. Besides, atomic operations are utilized to determine the minimum distance. The implementation details above are not revealed in our pseudo-code.