

Physically Embodied Gaussian Splatting: A Realtime Correctable World Model for Robotics

Supplementary Material

Anonymous Author(s)

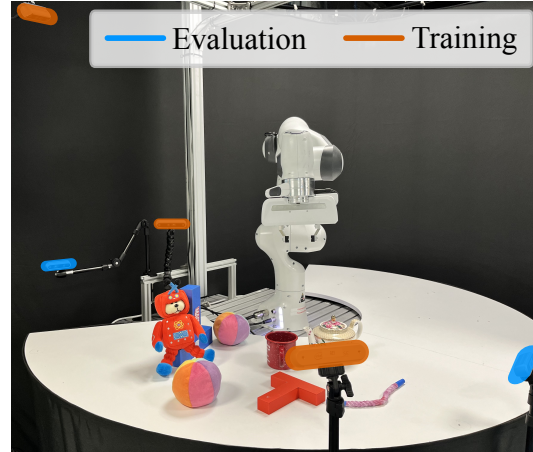
Affiliation

Address

email

A Experimental Setup

The real-world experiments are conducted using the tabletop setup shown in Suppl. Figure 1. The setup employs a Franka Emika robot equipped with two end-effectors: a standard gripper for pick-up scenarios and a pusher for other scenarios. The tabletop and robot are observed by five cameras: three Intel RealSense D455 cameras and two D435 cameras. These cameras are jointly calibrated using a hand-eye calibration technique. During operation, all five cameras are utilized for system initialization. However, only the three D455 cameras are employed during the prediction and correction stages. In all scenarios, the robot is teleoperated to manipulate objects on the tabletop. The datasets are captured by recording the image stream from the cameras and encoding them as HEVC videos. These videos are scaled to a resolution of 640x360 and decoded in real-time during evaluation to mimic live operation. Additionally, the robot's joint positions are timestamped and saved during the recording process and replayed during the evaluation.



Supplementary Figure 1: The tabletop setup used in the real experiments showing the robot, some of the objects used in the scenarios, and the position of the 5 cameras used.

B Implementation

The system follows a two-step process: initialization and prediction/correction. During initialization, particles and Gaussians are generated for each detected object in the scene. Subsequently, the system enters the prediction and correction stage, where the particles are simulated using a Position-Based Dynamics (PBD) physics system, while corrective forces are calculated based on the Gaussians attached to the particles. This section elaborates on the implementation and parameterization details of each phase.

Static Scene Initialization The tabletop is modeled using the five RGBD cameras in the scene, employing the standard Gaussian Splatting technique. However, to avoid interference with object placed on the table, the Gaussians are initialized as thin disks. Additionally, the table's pointcloud is utilized to calculate the ground plane. The Gaussians are trained using the Adam optimizer for 500 steps, with a position learning rate of $1e^{-4}$, color learning rate of $2.5e^{-3}$, scaling learning rate of $1e^{-3}$, opacity learning rate of $1e^{-2}$, and rotation learning rate of $1e^{-3}$. The scale is clamped between 1 mm and 1 cm.

Robot Initialization The robot’s particles are manually fitted to the links using Blender. The link each particle belongs to is stored so that forward kinematics can be used to appropriately change its position. Furthermore, the robot is rendered in Blender from multiple viewpoints, and Gaussians are trained to reconstruct these renders. These Gaussians are then bonded to the closest particle on the robot. The combination of particles, Gaussians, and bonds is inserted at the start of every scenario. The same parameters used for training the static scene are also applied to the robot.

Object Initialization For each object, the 3D bounding box is calculated from its point-cloud, which is extracted from the depth and instance masks. The initialization process is described in Algorithm 1. We use $n = 80$ and $m = 250$. All particles are initialized to a mass of 0.1 kg with the exception of the real and simulated rope which are set to 0.2 kg and 0.3 kg. The higher the mass of the particle, the less the influence of the visual force. The mass acts as both a *physical* and a *visual* inertia. These concepts can be separated in future work if fine-grained tuning is needed. For scenarios involving rope, the corrective forces are less reliable than that of larger bodies because they occupy less pixels in the image and the physical priors are less constraining because of the deformability. We compensate for the increased noise in the corrective forces by increasing the visual inertia. Note that Algorithm 1 is repeated for each object. Future work may choose to build all the objects simultaneously rather than sequentially to reduce the overall duration of the initialization. In the current implementation, object modeling takes approximately 20 to 40 seconds, which we found acceptable given that it is only done once per scenario.

Prediction Step The Position-Based Dynamics (PBD) physics system is used to predict the locations of the particles and the Gaussians at each timestep. It runs at a fixed frequency of 30 Hz (33.33 ms per step). The physics step is described in Algorithm 2. We use 20 substeps. At each substep, the velocities and forces are integrated, and then the constraints are solved using a Jacobi solver. Four Jacobi iterations are employed to sufficiently solve the physical constraints. After every physics step, the particle velocities are multiplied by 0.9 (an empirically chosen value). This damping contributes to system stability.

Algorithm 1 Dual Gaussian-Particle Initialization

```

1: Fill BBox with Spherical Gaussians
2: Prune Gaussians Not In Instance Masks
3: for n iterations do
4:   for all images and masks  $I$  do  $\triangleright$  Adam step
5:      $L \leftarrow L_{\text{rgb}} + L_{\text{seg}}$ 
6:      $L.\text{backward}()$ 
7:      $g, a, c = \text{optimizer.step}()$ 
8:   for k iterations do  $\triangleright$  Jacobi step
9:      $g = \text{solveCollisionConstraints}(g)$ 
10:     $g = \text{solveGroundConstraints}(g)$ 
11:   $p = g$   $\triangleright$  Create particles at Gaussian locations
12:  Initialize particle mass and velocities
13:  Create particle shape constraints
14:  for m iterations do
15:    for all images and masks  $I$  do
16:       $L_{\text{rgb}}.\text{backward}()$ 
17:       $g, a, c, s = \text{optimizer.step}()$ 
18:       $g, a, c, s = \text{densify}(g, a, c, a)$ 
19:    for each Gaussian  $i$  do
20:       $g_i.\text{parent} = \text{findClosestParticle}(p)$ 

```

Algorithm 2 PBD Physics Step

```

1: for all particles  $i$  do  $\triangleright$  Integrate particles
2:    $p_0, q_0 \leftarrow p_i, q_i$ 
3:    $p_i \leftarrow p_i + \Delta t v_i + \frac{\Delta t^2}{m_i} (f_i + \text{gravity})$ 
4:    $\theta \leftarrow \frac{|w_i| \Delta t}{|\omega_i|}$ 
5:    $q_i \leftarrow [\frac{2}{|\omega_i|} \sin \theta, \cos \theta] q_i$ 
6: for k solver iterations do  $\triangleright$  Resolve constraints
7:   for all particles  $i$  do
8:      $p_i \leftarrow \text{groundConstraints}(i)$ 
9:   for all collision pairs  $i, j$  do
10:     $p_i \leftarrow \text{collisionConstraints}(i, j)$ 
11:   for all shapes  $s$  do
12:     for particles  $i$  in  $s$  do
13:        $p_i, q_i \leftarrow \text{shapeMatching}(i, s)$ 
14: for all particles  $i$  do  $\triangleright$  Update velocities
15:    $v_i \leftarrow (p_i - p_0) / \Delta t$ 
16:    $\omega_i \leftarrow \text{axis}(q_i q_0^{-1}).\text{angle}(q_i q_0^{-1}) / \Delta t$ 

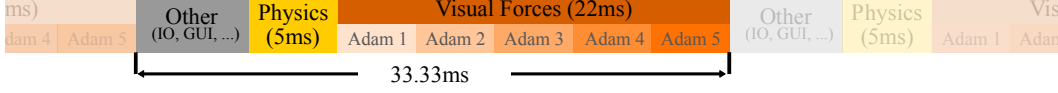
```

Algorithm 3 Visual Forces

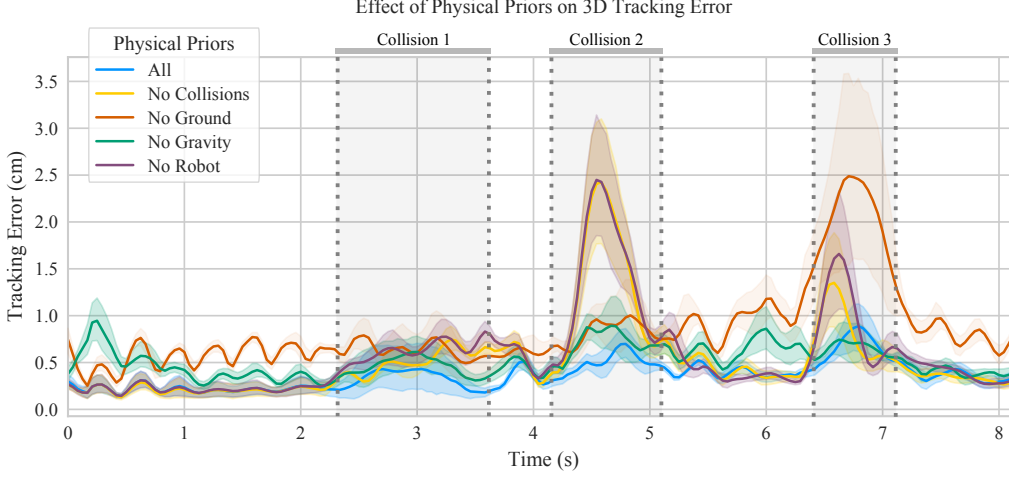
```

1:  $g^{\text{prev}} \leftarrow g$   $\triangleright$  Save positions
2:  $o = \text{AdamOptimizer}()$ 
3: for n iterations do
4:   Choose random image  $I$ 
5:    $L_{\text{rgb}}(I).\text{backward}()$ 
6:    $g[\text{not objects}].\text{grad} = 0$ 
7:    $g, c, o, R \leftarrow o.\text{step}()$ 
8: for every Gaussian  $i$  do
9:    $k = g_i.\text{parent}$ 
10:  if  $k$  is not None then
11:     $f_k \leftarrow f_k + K_p (g_i - g_i^{\text{prev}})$ 
12:   $g \leftarrow g^{\text{prev}}$   $\triangleright$  Reset positions

```



Supplementary Figure 2: The various functions called during the prediction and the correction step are profiled. In the ‘Other’ phase, the GUI is drawn and new sensor observations are read. The physics step takes 5 ms and is followed by approximately 22 ms of Adam optimizations that are used to compute the visual forces.



Supplementary Figure 3: An ablation showing the effect of different physical priors on the 3D tracking error of 12 points located on two objects on a tabletop. The scene used for this ablation is “Multiple1” from the simulated dataset. Using all physical priors produces on average the lowest tracking error over time.

Correction Step Visual forces are computed in the correction step using Algorithm 1. Gaussian displacements are calculated using 5 iterations of the Adam optimizer. The scales of the Gaussians are fixed, while the positions, rotations, opacities, and colors are allowed to change. Adam’s internal parameters are reset at every new physics step. Gaussian displacements below 2 mm are ignored to increase stability. The position learning rate is set to $1e^{-3}$, while the rotation, color, and opacity learning rates are set to $1e^{-4}$, $5e^{-4}$, and $5e^{-4}$, respectively. Allowing colors, opacities, and rotations to change gives the system more ways to explain lighting variations that should not be explained by the motion of the Gaussians. A K_p of 60 is used in all scenarios.

The prediction and correction steps are profiled in Suppl. Figure 2.

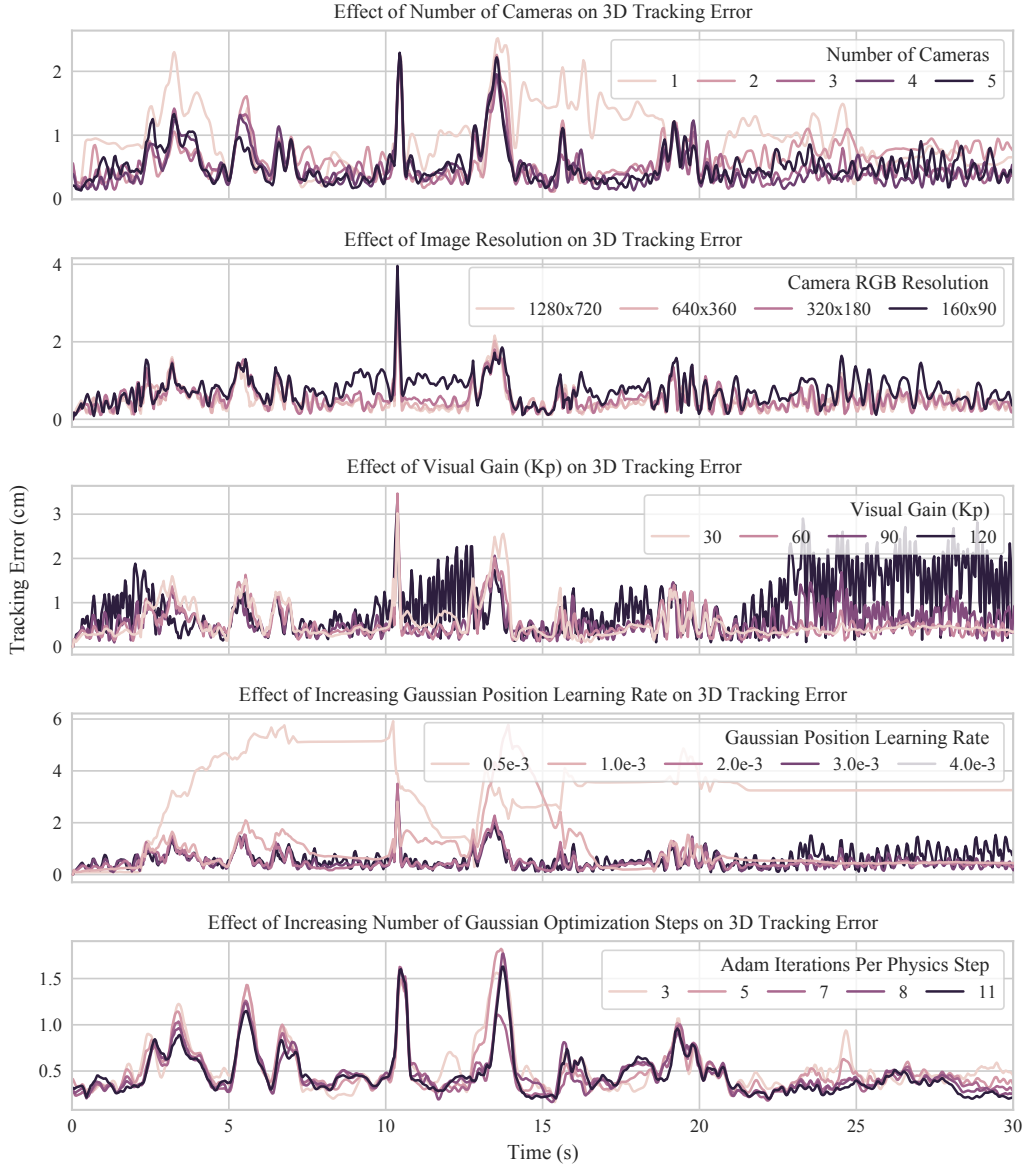
C Ablations

Physical Priors We evaluate the effectiveness of our system’s embedded physical priors by simulating a scene with two objects, as illustrated in Suppl. Figure 3. The scenarios highlight how our system’s performance is enhanced by incorporating various physical constraints: (i) With all physical priors enabled, our system accurately captures the objects’ dynamics, including collisions and interactions with the environment. (ii) When collisions between particles are ignored, the objects’ states deviate from the ground truth, particularly during intense collision events (Collisions 2 and 3). (iii) Disabling the ground plane and gravity causes the objects’ motions to oscillate continuously, as their movements are no longer properly regulated. (iv) Even with the ground plane intact, disabling gravity leads to similar oscillatory behavior, as the objects are not subjected to the expected downward force.

By adding physical priors, our system achieves better predictions that more closely match the groundtruth.

118 Suppl. Figure 4 ablates (i) the number of cameras used to compute visual forces, (ii) the resolution of
 119 the images used for the reconstruction loss, (iii) the effect of visual gain, (iv) the Gaussian position
 120 learning rate, and (v) the number of Adam iterations.

121 **Cameras** The ablations reveal that increasing the number of cameras yields diminishing returns
 122 within our framework. We observe that higher resolutions lead to lower tracking error. There is,
 123 however, only a slight difference between 1280x720, 640x360, 320x180. 1280x720 comes at a
 124 significant computational cost with visual force computation taking approximately 40 ms compared
 125 to the 20 ms for the lower resolutions. Below 640x360, the factor limiting performance is no longer
 126 resolution and thus there is no performance gain. For these reasons, we choose the 640x360 as the
 image size with which we calculate the visual forces.



Supplementary Figure 4: The effect of varying the parameters of our system on 3D tracking performance.

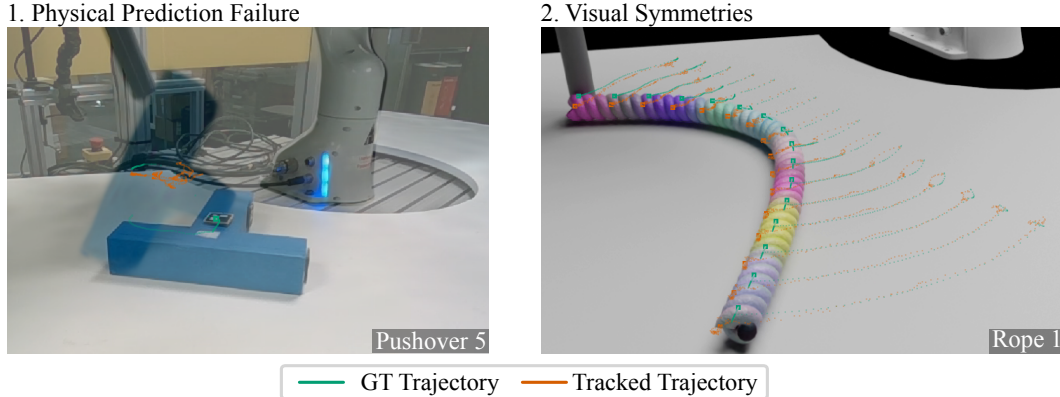
128 **Visual Forces** Our framework uses visual forces to create the corrective actions needed to keep the
 129 Gaussian-Particle representation synchronized. This results in smooth corrections but it also creates
 130 dynamic effects that, without careful tuning, creates oscillations. These oscillations are similar to
 131 the behaviour of an undamped spring system. Future work may look into removing the oscillatory
 132 effect by possibly adding a derivative term to the visual force calculation. For this work, we tune our
 133 system and find a balance between an acceptable amount of oscillations and tracking ability. The
 134 ablations in Suppl. Figure 4 show that a high gain (and/or a high Gaussian position learning rate)
 135 produces high oscillation and that a low gain (and/or a low learning rate) has a detrimental effect on
 136 tracking. The number of Adam iterations is chosen so that realtime constraints are met. The ablation
 137 shows that reducing the number of Adam iterations is a trade off that can be made when the physics
 138 timestep takes longer than expected without a significant impact on the overall synchronization of
 139 the world model.

140 D Failure Modes

141 The Gaussian-Particle representation can deviate from the groundtruth in several ways. If the ren-
 142 dered state of the scene significantly differs from the groundtruth image, the visual forces will not
 143 create meaningful corrections.

144 Additionally, if the physical modelling is significantly different to its real world counterpart, the
 145 physical priors will have a detrimental effect on the tracking performance of the system. This can be
 146 seen in the real scenario title “Pushover 5” in Suppl. Figure 5 where a T-Block could not be pushed
 147 over and thus escaped the radius of convergence of the visual forces.

148 In some instances, both the texture and the geometry of the object are simultaneously ambiguous.
 149 In the simulated “Rope 1” scenario, the rope can rotate around its spine without impacting either the
 geometry or the texture thus allowing for a slight steady state error to occur.

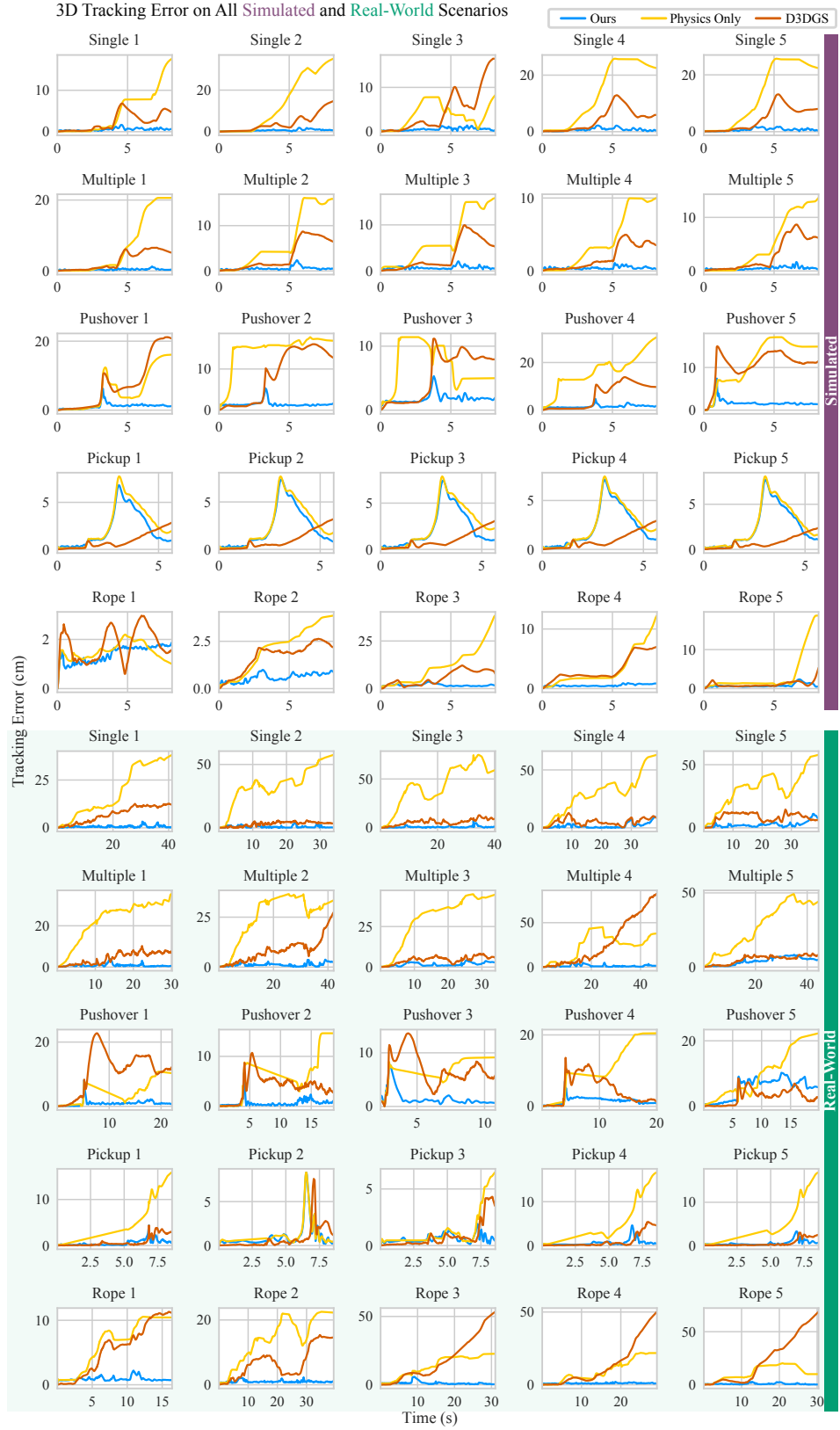


Supplementary Figure 5: The first image shows a highly dynamic scenario where the physics failed to push the TBlock into a location where visual forces could correct it. The second image shows a scenario where both visual and geometrical symmetries allowed the rope to rotate around its central axis and created a steady state error in tracking.

150

151 E Experimental Results

152 The 3D tracking performance of our system on all scenarios are shown in Suppl. Figure 6.



Supplementary Figure 6: The 3D tracking performance of our system and its baselines on all scenarios (simulated and real)