Collision Simulation with Image Input

Goals and Audience

Goal: Build an end-to-end system that ingests two images, segments the primary objects, infers approximate physical/material properties, reconstructs 3D meshes, and simulates a physically plausible collision with deformation, friction, bounce, and damping in Blender, producing a labeled animation and reportable parameters.

Audience: Technical evaluators in computer vision, graphics, and simulation; product stakeholders assessing feasibility of automated physics previews; and peers seeking a reproducible pipeline integrating segmentation, image-to-3D, and Blender's rigid body engine

Pipeline Design

Segmentation: Use a promptable instance segmentation model for zero-shot segmentation, enabling point/box prompts and multi-mask selection to isolate two foreground objects from each input image.

Material property inference: Map visual cues and class labels to a material prior library (density, restitution, friction, damping, stiffness) and expose tunable sliders; properties initialize Blender rigid body fields (mass, friction, restitution, damping) and constraints.

3D reconstruction: Generate watertight proxies from each segmented crop using TripoSR (fast, feed-forward image-to-3D following LRM principles) to obtain meshes suitable for collision simulation.

Scene/physics: Import meshes into Blender; set scale, origin, collision shape (mesh/convex hull), rigid body type (Active/Passive), mass, friction, restitution, linear/angular damping, collision margin, and constraints as needed; configure RigidBodyWorld parameters and bake.

Animation: Initialize positions and velocities to guarantee impact; optionally add randomness, stickiness (low restitution + higher friction), or permeability approximations via collision margin/collections and custom constraints.

Output: Render a short animation and a metadata panel with per-object properties (mass, bounciness, friction, damping), segmentation masks, and reconstruction thumbnails

Baseline Results

Reconstruction: TripoSR produces coherent single-image 3D assets in under a second on high-end GPUs, adequate for rigid-body proxy collisions.

Physics: Blender rigid body fields set per object (mass, friction, restitution, linear/angular damping, collision margin) yield plausible collisions under gravity with correct bounce/friction response.

Results of at least 1 development iteration from your baseline

Problem: Overly elastic collisions and tunneling on thin geometries; inconsistent scale caused unrealistic mass/energy behavior.

Changes:

Switched collision shapes to convex hull or mesh with nonzero collision margin and increased simulation substeps to reduce tunneling.

Normalized object scale and derived mass from estimated volume×density prior; tuned linear/angular damping for energy decay.

Outcome: More stable contacts, reduced jitter, and bounce consistent with specified restitution; collisions visually matched expected frictional sliding vs sticking modes

Video: https://drive.google.com/file/d/1i9W2ARJlutSUnPjWktT-PO5sIhq5D3tV/view?usp=sharing

In our blender script, we apply properties as follows:

```
robot.dimensions = (0.6, 0.54, 0.75)  # width, depth, height (in meters)
robot.rigid_body.mass = 0.58
robot.rigid_body.friction = 0.4
robot.rigid_body.restitution = 0.2  # bounciness
robot.rigid_body.use_margin = True
robot.rigid_body.collision_margin = 0.005

# Damping (for smoother motion)
robot.rigid_body.linear_damping = 0.8
robot.rigid_body.angular_damping = 0.05
```

We then apply a force to each object to simulate a collision:

```
obj = objs[0]
# Make object kinematic for first frame
obj.rigid_body.kinematic = True
obj.keyframe_insert(data_path="location", frame=1)
# Move object forward (simulate velocity) at next frame
obj.location.x += 2  # this is the “push” distance
obj.keyframe_insert(data_path="location", frame=15)
```

Plan for improvements

Material priors: Expand category-to-physics mappings with curated tables (density, Poisson's ratio, Young's modulus) and fit Blender parameters (mass, restitution, friction, damping) via heuristic regression.

Soft effects: Approximate deformation by blending rigid-body motion with shape keys or modifiers; where feasible, selectively use soft body or constraints for bending modes.

Interactive: Letting the user customize some properties of the objects and collision environment.

Reconstruction: Integrate native Tripo SDK or Blender add-on for faster round-trips; evaluate alternative Tripo family models for topology-sensitive objects.

Robustness: Auto-detect and fix mesh issues (non-manifold, inverted normals) and auto-scale to meter units to stabilize solver behavior

Timeline, milestones, member duties

Phase	Milestone	Vivasvat	Darren	Kyle
Immediate (now–Oct 30)	**Complete v1 feature build**	UI for property editing	Add new collision properties	API for image-to-properties
By Oct 31	**Submit Midterm Report**	Contribute UI + doc	Contribute pipeline, properties + doc	Contribute API, help with report
Nov 1–Nov 15	**2nd iteration and website exploration**	Web UI draft + test	Backend eval + deployment test	Blender-to-web pipeline prototype
End of Term (Final week)	**Integrate, polish, & submit deliverables**	UI finalization, compile docs	Pipeline stable, asset pipeline stable	Export script, code packaging