Rigid Body Collision : A Project Report

Pratyay Dutta, Aditya Gambhir
(Contributed Equally)

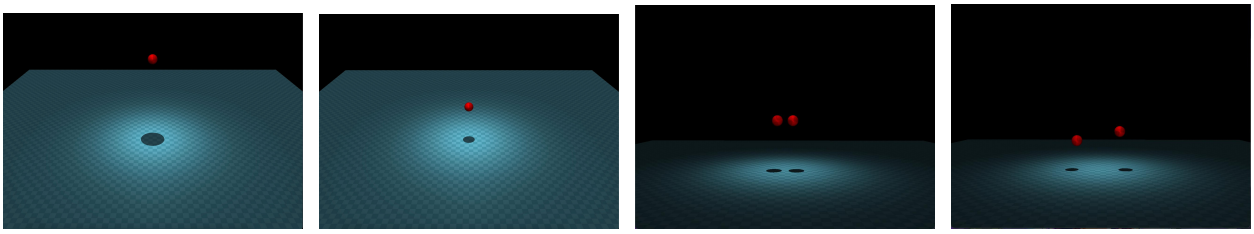
March 19, 2025

Abstract

This report presents the implementation of a rigid body collision simulation using a time integration scheme and a collision response scheme from the seminal work of Guendelman et. al [1]. We demonstrate time-stepped integration, collision detection, and collision response with the ground and an additional rigid ball. Our approach includes computing impulses, applying a coefficient of restitution, and updating the velocity/position states of the rigid bodies.

1 Introduction

The simulation of rigid body collisions has been a central topic in computer graphics, robotics, and computational mechanics for decades. Early attempts at rigid body collision simulation often relied on penalty-based methods, where colliding bodies were given large restoring forces that pushed the objects apart. While these approaches were conceptually straightforward, they could lead to numerical stiffness, requiring very small time steps to avoid instability or undesired visual artifacts.



(a) (b) (c) (d)

Figure 1: Experimental demonstration of our implementation. (a) Ball to ground initial position. (b) Ball to ground after collision. (c) Ball to ball before collision. (d) Ball to ball after collision.

Another popular line of work involved constraint-based or impulse-based methods, which computed collision impulses by formulating velocity-level constraints at the contact points. These methods were effective in handling single collisions or simple stacking scenarios but

frequently struggled with simultaneous collisions or stacking of multiple rigid bodies. In particular, naive implementations could produce jitter, penetrations, or indefinite bouncing when resting contact was not handled robustly.

Guendelman et al. [1] proposed a novel time-sequencing scheme for handling collisions, interpenetration, and stacking in a unified manner. Their approach tackled the common pitfalls of:

- **Indefinite Bouncing:** Standard solvers might apply a restitution impulse on objects even when they should be at rest, leading to perpetual micro-bounces.
- **Inexact Contact Resolution:** Many iterative methods require ad hoc thresholds or repeated solver sweeps that can be computationally expensive and produce unpredictable results.
- **Stacking Instability:** Large stacks of objects tended to either collapse or jitter when collisions were only partially resolved, particularly when the contact forces were not updated consistently before the next integration step.

By sequencing collision detection and response in a more carefully orchestrated manner, and by integrating velocities and positions at moments that avoid applying impulses on objects already at rest, the Guendelman scheme achieved greater stability and realism. This has made it influential in the development of modern physics engines for both graphics and robotics research.

In this work, we implement the core ideas of the Guendelman time-sequencing method atop the MuJoCo simulation framework [2]. While MuJoCo itself already offers robust physics for a wide range of robotic applications, our goal was to directly integrate and validate the Guendelman schema on a familiar simulation platform, ensuring that we retain the conceptual advantages of their approach. Specifically, we:

- Injected a custom time-stepped integration scheme into MuJoCo's simulation loop.
- Incorporated impulse-based collision detection and response with configurable coefficients of restitution.
- Provided placeholders for future expansions such as friction, complex geometries, and multi-contact scenarios.

In the following sections, we discuss the motivation behind our project (Section 2), the implementation details (Section 3), and key experimental demonstrations (Section 4). We conclude by summarizing the lessons learned and proposing future extensions (Section 5).

2 Motivation

Physics-based simulations allow researchers and developers to test and iterate on control strategies, robot designs, and interactive environments without the expense of real-world experimentation. This project:

- Investigates how to integrate a custom time-stepped solver into MuJoCo's environment.

- Explores impulse-based collision response with ground and ball-to-ball contacts.
- Provides a basis for future expansion to more complex shapes, friction models, and constraints.

3 Implementation

The implementation can be divided into three major components: the custom time-step integration, collision detection, and collision response.

Time-Stepped Integration

We adopt a semi-implicit Euler scheme, where velocities are updated before positions. In pseudocode:

```
1. for each timestep:
2.    apply external forces to update velocity:
         v_{new} = v_{old} + (F/m)*dt
3.    detect and handle collisions (see next subsections)
4.    update positions using the new velocities:
         x_{new} = x_{old} + v_{new} * dt
```

This approach helps ensure that collisions are handled with up-to-date velocities, mitigating jitter and providing stable contact.

Collision Detection

Since we focus on collision with the ground plane and with another spherical ball:

- **Ground Collision:** We check if the lowest point of the ball is below or at the ground $z = 0$. If the ball's center is at z_{center} and radius is r, collision occurs if $z_{center} - r \leq 0$.
- **Ball-to-Ball Collision:** Two balls collide if the distance between their centers is less than or equal to the sum of their radii. For centers $\mathbf{x}_1, \mathbf{x}_2$ and radii r_1, r_2:

$$\|\mathbf{x}_1 - \mathbf{x}_2\| \leq (r_1 + r_2).$$

Collision Response

We use an impulse-based method to resolve collisions. Suppose two colliding rigid bodies A and B have velocities $\mathbf{v}_A, \mathbf{v}_B$ (linear) and angular velocities $\boldsymbol{\omega}_A, \boldsymbol{\omega}_B$. The collision impulse $\mathbf{j}$ along the collision normal $\mathbf{n}$ (pointing from A to B) is computed as:

$$\mathbf{j} = -(1+\epsilon)\frac{\mathbf{v}_{rel} \cdot \mathbf{n}}{(\frac{1}{m_A} + \frac{1}{m_B})},$$

where ϵ is the coefficient of restitution and $\mathbf{v}_{rel}$ is the relative velocity at the contact point. For ball-to-ground collisions, treat the ground as having infinite mass.

Once $\mathbf{j}$ is found, update velocities:

$$\mathbf{v}_A \leftarrow \mathbf{v}_A + \frac{\mathbf{j}}{m_A}, \quad \mathbf{v}_B \leftarrow \mathbf{v}_B - \frac{\mathbf{j}}{m_B}.$$

If angular velocities are considered (for non-spherical shapes or frictional collisions), incorporate torque about the contact point.

Friction Model

Friction is an essential component in rigid body dynamics, ensuring realistic contact interactions by preventing excessive slipping. In our implementation, we incorporate both static and kinetic friction using an impulse-based approach, where frictional forces are resolved through a tangential impulse acting opposite to the relative velocity at the contact point.

For a rigid body in contact at a point $\mathbf{p}$, the relative velocity at the contact point is given by:

$$\mathbf{v}_{rel} = (\mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{r}_2) - (\mathbf{v}_1 + \boldsymbol{\omega}_1 \times \mathbf{r}_1), \tag{1}$$

where $\mathbf{v}_1, \mathbf{v}_2$ are the linear velocities, $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2$ are the angular velocities, and $\mathbf{r}_1, \mathbf{r}_2$ are the contact position vectors relative to the center of mass of each body.

This velocity is decomposed into normal and tangential components:

$$\mathbf{v}_{rel} = \mathbf{v}_{rel,n} + \mathbf{v}_{rel,t}, \tag{2}$$

where the normal component is:

$$\mathbf{v}_{rel,n} = (\mathbf{v}_{rel} \cdot \mathbf{N})\mathbf{N}, \tag{3}$$

and the tangential component is:

$$\mathbf{v}_{rel,t} = \mathbf{v}_{rel} - \mathbf{v}_{rel,n}. \tag{4}$$

The normal impulse j_n is computed using the coefficient of restitution ϵ:

$$j_n = -(1+\epsilon)\frac{\mathbf{v}_{rel} \cdot \mathbf{N}}{\mathbf{N}^T(\mathbf{K}_1 + \mathbf{K}_2)\mathbf{N}}, \tag{5}$$

where $\mathbf{K}_i$ represents the inverse mass-inertia matrix at the contact point. The frictional impulse, acting in the tangential direction, is given by:

$$j_t = -\min(\mu j_n, |\mathbf{v}_{rel,t}|)\frac{\mathbf{v}_{rel,t}}{|\mathbf{v}_{rel,t}|}, \tag{6}$$

where μ is the coefficient of friction and j_n is the normal impulse. This equation ensures that the frictional impulse adheres to Coulomb's law, where it does not exceed μj_n but still acts to reduce the relative tangential velocity.

After computing the normal and tangential impulses, we update the velocities of the rigid bodies as:

$$\mathbf{v}_i' = \mathbf{v}_i + \frac{j_n \mathbf{N} + j_t}{m_i}, \tag{7}$$

$$\boldsymbol{\omega}_i' = \boldsymbol{\omega}_i + \mathbf{I}_i^{-1}(\mathbf{r}_i \times (j_n \mathbf{N} + j_t)). \tag{8}$$

Here, $\mathbf{I}_i$ is the inertia tensor, and $\mathbf{r}_i$ is the contact position relative to the center of mass. Static and kinetic friction are handled based on the magnitude of the tangential velocity:

- If $|\mathbf{v}_{rel,t}| = 0$, static friction is applied, preventing motion.
- If $|\mathbf{v}_{rel,t}| > 0$, kinetic friction acts to reduce tangential motion following the impulse-based formulation.

This impulse-based friction model allows for stable and realistic simulation of rolling, sliding, and dynamic contact interactions.

4 Experiments

4.1 Experimental Setup

We ran experiments with two spheres (each of radius $r = 0.1$ m).

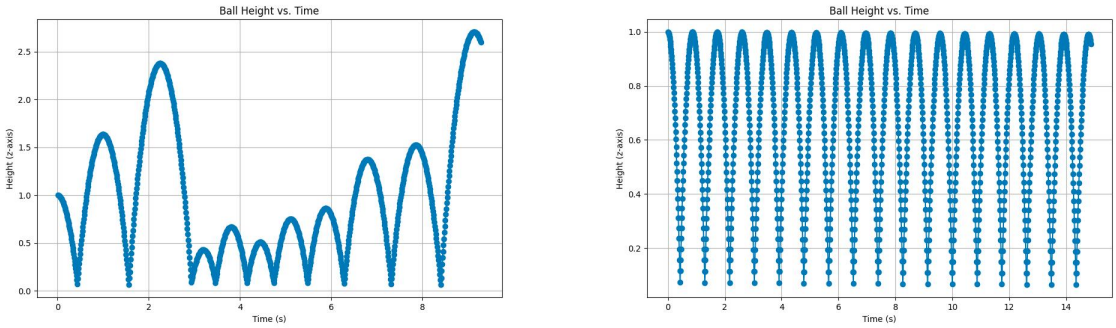
- **Ball to Ground**: Ball is dropped from a height of 0.2 m from the ground.
- **Ball to Ball**: Two identical balls are given an initial linear velocity towards each other.

One sphere is dropped from a height of 2 m to test ground collision, while the other sphere is initially placed at rest to test sphere-sphere collision once the first sphere impacts and bounces. Our experiments:

- elastic collision height graph before and after time integration.
- elastic collision before and after impulse algo implementation.

4.2 Ball to Ground collisions

In the case of inelastic collision, due to the dampening effect of the frictional force acting normal to the surface, the time integration and the collision algorithm does not make a lot of difference in that case. However, for elastic collisions we see a lot of difference. Before the time integration algorithm, as explained in Section 1, there were a lot of artefacting due to error buildup and microcollisions. The ball shoots up after a while and comes back down. However, after the time integration scheme, since the position is being estimated beforehand and the velocity of the ball is rectified accordingly, we see that we can sustain elastic collision for a long time. This is demonstrated in Figure 2



(a) (b)

Figure 2: Comparison of elastic collision before and after time integration. (a) Before time integration, we can see that due to artefacting and the hallucinations from microcollisions, the bouncing is very inconsistent. (b) After time integration, the collisions are consistent and maintains elasticity property of undampened collision.

After we implemented the tangential frictional and added it to the impulse calculation method, we initiated our sphere with an angular velocity of $0, -0.5, -.5$ (x,y,z). We tracked the trajectory of the ball after initialisation and it is shown in Figure .. This verifies that our frictional model works as the ball spins after collision with the ground.

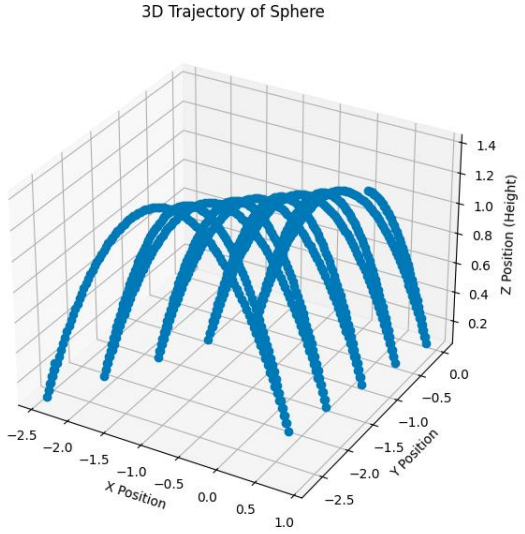


Figure 3: Trajectory of ball after tangential friction modeling. Initial linear velocity = $(2, 0, 0)$. Initial Angular velocity = $(3, 0, 0)$

4.3 Ball to Ball Collision

We give an initial linear velocity to both the balls towards each other so that they collide. We can see that our modeling paradigm follows real world physics based movements just like we would expect. The trajectory of the balls are shown in Figure 4. In this experiment, all the functions talked above are implemented.

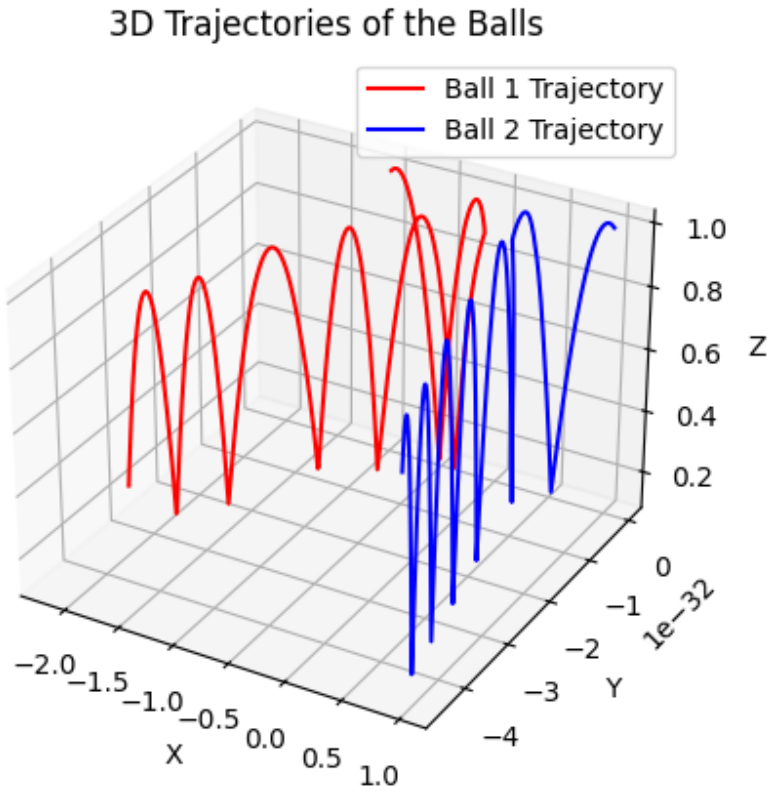


Figure 4: Trajectory of two balls colliding against each other.

5 Conclusion

We successfully implemented a rigid body simulation with collisions using a custom time-stepped integration and impulse-based collision response. The key contributions include:

- Adapting MuJoCo's callback system to inject custom physics.
- Demonstrating stable collision handling for ball-to-ground and ball-to-ball contact.

Future directions include:

- Extending collision detection to complex geometries with bounding boxes or convex hulls.

- Investigating performance optimizations for larger-scale simulations.

References

[1] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM transactions on graphics (TOG)*, 22(3):871–878, 2003.

[2] Emanuel Todorov and Igor Mordatch. Mujoco documentation: Advanced interactive robot simulation, 2018. https://mujoco.org/book.