BagSim: A Realistic Physics Simulator for Deformable Bags

Lawrence Yunliang Chen
University of California, Berkeley
USA
yunliang.chen@berkeley.edu

Karthik Dharmarajan
University of California, Berkeley
USA
kdharmarajan@berkeley.edu

Andy Huang
University of California, Berkeley
USA
handy1221@berkeley.edu

Jeffrey Tan
University of California, Berkeley
USA
tanjeffreyz02@berkeley.edu

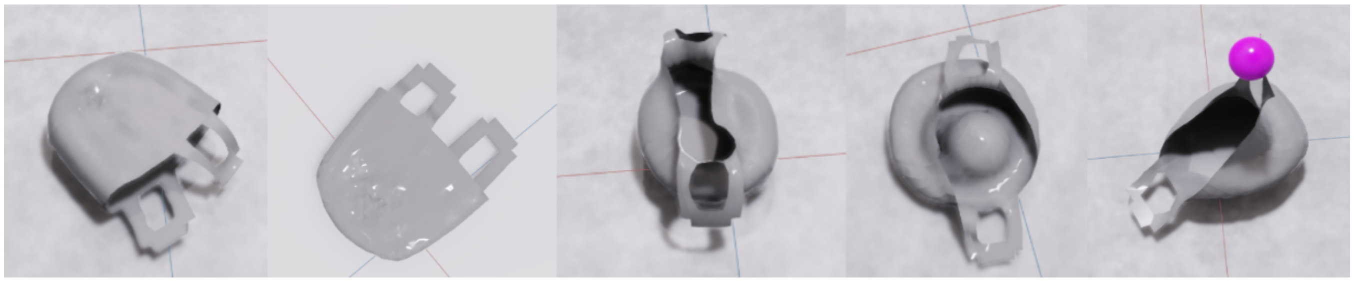

Figure 1: Sample images of a bag in Nvidia Omniverse. First three images show the bag at various rest states after being perturbed. Fourth image is the bag sitting on top of a ball. The last image shows one bag handle attached to a floating sphere.

ABSTRACT

Simulating the behavior of deformable objects is a challenging task in computer graphics but has wide applications including animation, virtual reality, and gaming. In this project, we focus on simulating deformable bags. Simulating deformable bags is particularly difficult due to their complex shapes, varieties of materials, and complex behaviors for self-collision and collision with other objects. Existing simulators use simplifications such as pretending the bags to be a piece of folded fabric, and fail to accurately simulate critical properties of bags such as handles and different bag shapes. In this project, we present BagSim, a physics-based simulator built on NVIDIA Omniverse for simulating deformable bags. We plan to model the shapes of various types of bags (with and without handles or bottoms, stiff or soft fabric material), and simulate the bag movement (elasticity, dynamics, deformation) when dragged around by a cursor. We will simulate the interaction between the deformable bag and rigid objects when placed into the bag, and tune the simulation parameters to approximately match real bags manually. We will provide both visual and quantitative evaluation of the simulator across each of the different bag types (comparing BagSim to SoftGym, DeformableRavens, DEDO, and a similar real bag), and benchmark the computation time for generating the state of the bag after some perturbation.

CCS CONCEPTS

• **Computing methodologies → Physical simulation**; *Motion processing*; *Collision detection*; *Mesh geometry models*; Ray tracing.

KEYWORDS

Deformable object simulation, Plastic bag, Physical simulation

1 INTRODUCTION

Accurate simulation of deformable objects such as cables, cloth, fluid, and biological tissues is a crucial component in many applications such as animation, computer-aided design, films, and games. The demand for realistic motions becomes even more apparent for virtual reality applications and robotic simulators as the quality of simulation makes a huge difference in the user experience and for training machine learning models. However, it is also well-known that modeling and simulation of such deformable objects are challenging, and they have been an active research area in mechanical engineering and computer graphics for several decades.

There are many different ways to model the deformation of objects [7]. On the one extreme of the spectrum are non-physical approaches, where a designer manually adjusts control points or shape parameters to edit or design the shape. However, this is not scalable. On the other end of the spectrum are approaches that are based on continuum mechanics, which considers the material properties, external forces, and environmental constraints that affect the deformation of objects. However, tuning the parameters, also known as system identification, is also very time-consuming and challenging. Even with identified parameters, there are still aleatoric uncertainties in the real world that cannot be fully modeled. Additionally, simulating and rendering realistic visuals can be slow and inaccurate.

In recent years, there has been huge progress in the physics-based simulation of cables and fabrics, which has been found useful by the robotics community to learn control policies for deformable objects [8, 10, 19]. For example, OpenAI Gym [3] and Mujoco [15] both support rope and cloth simulations, as well as sponge-like soft 3D objects. Lin et al. [11] build a higher-fidelity simulator for ropes, cloth, and water. Huang et al. [9] simulate soft 3D objects such as fruits, internal organs, and flexible containers.

In this project, we focus on deformable bag simulation. There has been a growing interest in simulating the behavior of bags, which are ubiquitous in daily life, including in home, commercial, and industrial settings. Building a high-fidelity simulator can be useful in robotics as it can significantly reduce the real-world data required for training models. For example, [4, 5] study the task of opening a plastic bag and inserting objects, and they train their model entirely in real, which requires collecting a large amount of real-world data. This can potentially be accelerated with the help of a bag simulator. Similarly, [18] leveraged simulation to learn a robot policy for hanging a bat onto a rack. In contrast to cloth, however, there exist very few high-fidelity simulators for bags. DeformableRavens [14] is one of the first few works that present physics-based bag simulation. However, their bag model is hugely simplified. [16] presents a graph-based model of a handbag and its simulation. Antonova et al. [1] build up on that and allow more complex movements. Still, the bag model is limited and the parameters are not systematically tuned to match with the real world. We aim to address these shortcomings.

Simulating deformable objects like bags is challenging due to the variety of behaviors depending on the material used. Existing simulators fail to accurately simulate critical properties of bags such as handles and different bag shapes, which limits their usefulness for several computer graphics applications. In this work, we present BagSim, a physics-based simulator environment built on top of NVIDIA Omniverse for simulating deformable bags.

Our proposed approach is to model a bag as several attached cloths with adjustable stiffness parameters. We also incorporate air resistance into our simulations to more accurately model real-world interactions with bags. Our system can generate realistic animations of bags that can be used in several computer graphics applications such as animation, virtual reality, and gaming.

The project goal is to contribute BagSim, a realistic physics simulator environment for simulating the movement of deformable bags, a systematic way of tuning the parameters, and an experimental evaluation of BagSim compared to existing simulators as well as bags in the real world.

In particular, we plan to model the shapes of various types of bags (with and without handles or bottoms, stiff or soft fabric material), and simulate the bag movement (elasticity, dynamics, deformation) when dragged around by a cursor. We will simulate the interaction between the deformable bag and rigid objects when placed into the bag, and tune the simulation parameters to approximately match real bags manually. We will provide both visual and quantitative evaluation of the simulator across each of the different bag types (comparing BagSim to SoftGym, DeformableRavens, DEDO, and a similar real bag), and benchmark the computation time for generating the state of the bag after some perturbation.

2 RELATED WORK

2.1 Deformable Object Simulation

There are many ways for simulating deformable objects, including non-physical models, mass-spring models, particle models, continuum models, and finite element methods [7]. Although most models incorporate some physical principles to compute the shapes or motions of deformable objects, in many applications, particularly in design, employ purely geometric techniques[2]. This highly relies on the skill of the designer and is not scalable.

Mass-spring systems are a physics-based model that has been used widely and effectively for modeling deformable objects due to their simplicity [7]. In mass-spring systems, an object is modeled as a collection of point masses connected by springs. Using Newton's laws, a simple second-order differential equation can be derived, and the position of the masses can be simulated using numerical ODE solvers, such as implicit Euler's method and Verlet integration. Although they are simple to implement, mass-spring systems have some drawbacks. First, it is not very accurate as the discrete model is a significant approximation of the true physics that occurs in a continuous body. Second, it has poor stability in stiff systems, and extra damping needs to be added.

Continuum models treat deformable objects as a continuum: solid bodies with mass and energies distributed throughout [7]. The finite element method (FEM) is a widely-used approximation for a continuous function that satisfies some equilibrium expressions [13]. In FEM, the continuum, or object, is divided into elements joined at discrete node points. A function that solves the equilibrium equation is found for each element. While more accurate than mass-spring systems, FEM is significantly computationally heavier.

Many current state-of-the-art physics-bases simulator engines use particle-based systems, such as SoftGym [11], Isaac Gym [12], and NVIDIA Omniverse. Particle-based systems can be derived from continuum mechanics [6] and achieve a good balance between fidelity and speed. In this work, we leverage the PhysX engine of NVIDIA Omniverse and its particle-based simulation model for simulating deformable bags.

2.2 Modeling and Simulation of Cables, Fabrics, and Fluids

Researchers have developed several analytic physics models to describe the dynamics of moving cables. For example, Gatti-Bonoa et al. [15] present a 2D dynamic model of fly fishing. They model the fly line as a long elastica and the fly rod as a flexible Euler-Bernoulli beam, and propose a system of differential equations to predict the movement of the fly line in space and time. In contrast to continuum models, Wang et al. [57] propose using a finite-element model to represent the fly line by a series of rigid cylinders that are connected by massless hinges.

Prior work related to the simulation of deformable objects such as fabrics, cables, and fluids also focused on creating an artificial environment in which robots could be trained to manipulate such objects. [17] develops an efficient high-fidelity simulator for simulating fluids as they interact with rigid objects. [14] presents an open-source simulation and benchmark called *DeformableRavens* based on the PyBullet engine for robotic manipulation of deformable 1D, 2D, and 3D objects. It provides support for such deformable objects using soft-body physics based on mass-springs and self-collisions among vertices. While this approach to simulation is well-tested and simple, it is far from accurate. In [10], the authors study various simulator models for accurately simulating the movement of a cable, including a PyBullet model consisting of capsule rigid bodies connected by 6 DOF spring constraints, a soft-body rod connected to a capsule rigid body at the endpoint, and a segmented model also consisting of a string of capsule rigid bodies but is connected using ball joints. SoftGym [11] is one of the best simulators for simulating cloths, also using particle-based models. However, it does not support more complex fabric meshes as the fidelity rapidly decreases with complex self-collision modeling.

2.3 Simulation of Deformable Bags

DeformableRavens [14] is one of the first few works that offer physics-based simulation of bags. However, their bag model is a hugely-simplified mesh by wrapping a single piece of cloth so it does not have handles. Additionally, as its physics is based on mass-springs systems, the simulated material is far from accurate as they scrunch up, flatten out, form creases, and ultimately retain some of the changes in their structure due to external forces. Mass-spring systems like the ones used in *DeformableRavens* can not accurately reflect these properties of bags, which may cause robots trained using the simulation to behave in unexpected ways in real-world scenarios. Moreover, it can easily go unstable when the parameters of the material are beyond some ranges.

Weng et al. [16] presents a graph-based model of a handbag and its simulation. They design the bag templates with handles to include multi-hole structures using Blender, and build the simulation environment in Unity based on Obi Cloth extension. Antonova et al. [1] build up on that by using similar mesh models but use the Isaac Gym [12] environment to allow more complex movements. Still, the bag model is limited and the parameters are not systematically tuned to match with the real world. In this work, we leverage the particle-based model offered by the Omniverse PhysX engine to create more realistic simulations both visually and dynamically.

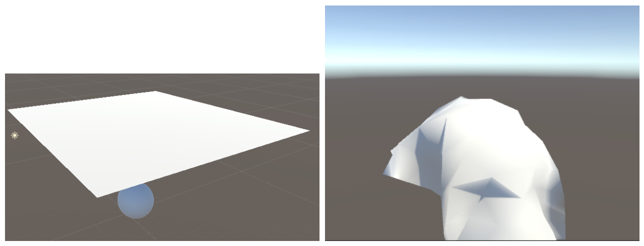
Figure 2: Default plane with cloth physics falls and deforms around a sphere in Unity.

3 SIMULATOR COMPARISON

3.1 Unity

In terms of scripting capability, out of the box Unity does not have much flexibility for customizing cloth physics through their UI or scripting API. Although, basic parameters for the entire cloth can be tweaked through both mediums, there ultimately is not a particle level scripting API offered by Unity alone which allow for the manipulation of cloth properties on a particle level. Ideally, to accurately capture behavior of bags of varying material and shape, we will need to assign varying cloth properties to different parts of a single cloth mesh. Unity applies properties across a cloth mesh

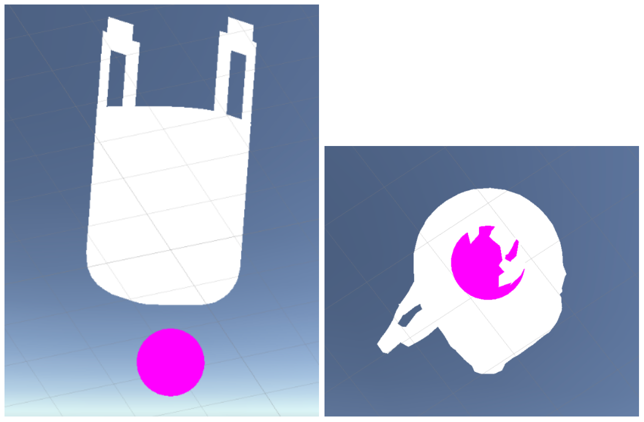
Figure 3: Custom bag mesh with cloth physics falls and deforms around a sphere in Unity.

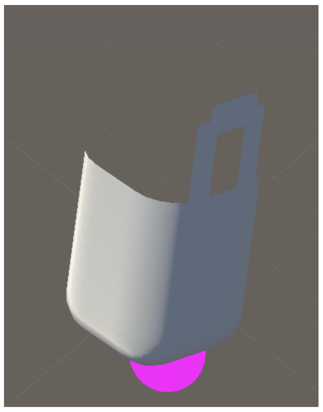
Figure 4: Imported bag mesh with a cloth component rendered into Unity.

uniformly. For example, varying cloth properties across a mesh is critical when modeling bags that have handles that are of a different stiffness and less stretchy than the body of the bag.

As an alternative, we have considered using the Obi Cloth Unity plugin and the accompanying Obi scripting API which we suspect can assign and manipulate cloth properties on a more granular level. However, we are hesitant to test it out ourselves because of the $47 fee to acquire it from the Unity asset store.

3.2 NVIDIA Omniverse

NVIDIA Omniverse has several capabilities that allow it to take in arbitrary meshes and convert them into deformable objects or particle based cloths, as shown in Figure 1. Using the default parameters when converting an arbitrary mesh to a cloth leads to very odd results. Upon some parameter tuning, however, the cloth can be made to look like a proper bag. NVIDIA Omniverse has a Python API, where anything from the user interface and be done if not more. Critically, this includes programatically setting the particle size, which is an important parameter that may need to be dynamically tuned. Another major benefit of Omniverse is that it uses

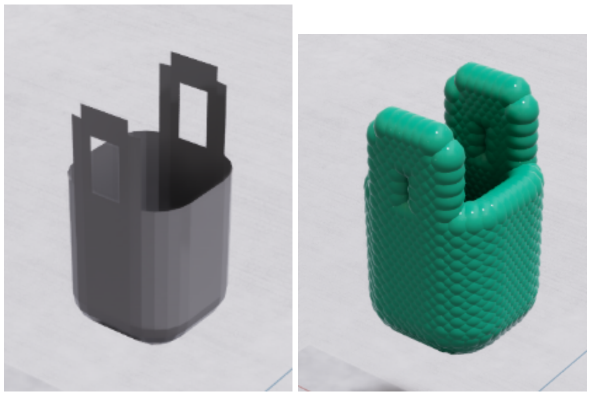

Figure 5: Mesh and particle views of a custom bag mesh with handles.

RTX for efficient and photorealistic ray tracing. There are plenty of options for creating the bag material, so the plastic finish for the bag can also be made. Another significant advantage of Omniverse is the ease at which collisions can be set up. In fact, Figure 1 shows the bag in collision with various objects in the scene, including the ground plane and a sphere.

Omniverse has several notable limitations, including the inability to program different properties per particle or different attributes per different portions of the cloth. Omniverse can only be run on Windows or Linux, limiting potential users.

4 BAG MODELS

To gain familiarity with the capabilities of Unity and Omniverse, we imported basic bag meshes into both simulators and setup a simple simulation of the meshes with cloth properties. The bag meshes that we imported into Unity and Omniverse are adapted from the DEDO Github repository linked in Dynamic Environments with Deformable Objects [1].

After loading a bag mesh into Unity, we ran into a problem where the outer surface of the bag mesh renders fine, however the inside is completely transparent. After a bit of debugging, we discovered that Unity performs a rendering optimization called "backface culling" where only one side of triangle meshes are rendered. The mesh faces that are rendered by Unity are the front faces. In other words, sides of the faces with normals pointing out of them. Thus, if we decide to continue working with Unity, we will likely need to create our own bag meshes in Blender that have double sided faces. On the other hand, when importing the bag mesh into Omiverse, both the front and back faces are rendered in.

Currently, our goal is to leverage Blender to create a bag mesh with a more realistic appearance, similar to the plastic bags commonly found in grocery stores.

5 PROGRESS SUMMARY

We have done extensive research on whether to use Unity or NVIDIA Omniverse as the platform we will build on top of. We have explored the capabilities of both these platforms by experimenting with their UIs to tweak cloth parameters. We will most likely need to further explore capabilities of their scripting API's to see if we can get more customizable cloth behavior that the UI simply does not offer.

While we are not as far along as we had outlined in our project proposal, the progress researching the different platforms is critical, and may save us time as we approach the project deadline. Since some of the features are able to be handled by the different platforms natively, we are planning on taking on other items, such as adding quantitative evaluation and more systematic tuning.

ACKNOWLEDGMENTS

We thank Professors Ren Ng and James O'Brian, and the CS 184 course teaching staff for their support and helpful feedback for this project.

REFERENCES

[1] Rika Antonova, Peiyang Shi, Hang Yin, Zehang Weng, and Danica Kragic Jensfelt. 2021. Dynamic environments with deformable objects. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[2] Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable object animation using reduced optimal control. In *ACM SIGGRAPH 2009 papers*. 1–9.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] Lawrence Yunliang Chen, Baiyu Shi, Roy Lin, Daniel Seita, Ayah Ahmad, Richard Cheng, Thomas Kollar, David Held, and Ken Goldberg. 2023. Bagging by Learning to Singulate Layers Using Interactive Perception. *arXiv preprint arXiv:2303.16898* (2023).

[5] Lawrence Yunliang Chen, Baiyu Shi, Daniel Seita, Richard Cheng, Thomas Kollar, David Held, and Ken Goldberg. 2022. AutoBag: Learning to Open Plastic Bags and Insert Objects. *arXiv preprint arXiv:2210.17217* (2022).

[6] Olaf Etzmuss, Joachim Gross, and Wolfgang Strasser. 2003. Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Transactions on Visualization and Computer Graphics* 9, 4 (2003), 538–550.

[7] Sarah FF Gibson and Brian Mirtich. 1997. *A survey of deformable modeling in computer graphics*. Technical Report. Technical report, Mitsubishi Electric Research Laboratories.

[8] Huy Ha and Shuran Song. 2022. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robot Learning*. PMLR, 24–33.

[9] Isabella Huang, Yashraj Narang, Clemens Eppner, Balakumar Sundaralingam, Miles Macklin, Tucker Hermans, and Dieter Fox. 2021. Defgraspsim: Simulation-based grasping of 3d deformable objects. *arXiv preprint arXiv:2107.05778* (2021).

[10] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. 2022. Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 8282–8289.

[11] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. 2021. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*. PMLR, 432–448.

[12] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. 2021. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470* (2021).

[13] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. 2006. Physically based deformable models in computer graphics. In *Computer graphics forum*, Vol. 25. Wiley Online Library, 809–836.

[14] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. 2021. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4568–4575.

[15] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[16] Zehang Weng, Fabian Paus, Anastasiia Varava, Hang Yin, Tamim Asfour, and Danica Kragic. 2021. Graph-based task-specific prediction models for interactions between deformable and rigid objects. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 5741–5748.

[17] Zhou Xian, Bo Zhu, Zhenjia Xu, Hsiao-Yu Tung, Antonio Torralba, Katerina Fragkiadaki, and Chuang Gan. 2023. FluidLab: A Differentiable Environment for Benchmarking Complex Fluid Manipulation. In *International Conference on Learning Representations*.

[18] Yifan You, Lin Shao, Toki Migimatsu, and Jeannette Bohg. 2021. Omnihang: Learning to hang arbitrary objects using contact point correspondences and neural collision estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5921–5927.

[19] Mélodie Hani Daniel Zakaria, Miguel Aranda, Laurent Lequièvre, Sébastien Lengagne, Juan Antonio Corrales Ramón, and Youcef Mezouar. 2022. Robotic Control of the Deformation of Soft Linear Objects Using Deep Reinforcement Learning. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, 1516–1522.