Intuitive Physics and Simulation Tools for Active Agents: A 2020 Draft Review

Rough draft research report (2020)

Ria Cheruvu

Note: This 2020 draft report surveys intuitive physics and general simulation tools for active agents, synthesizing AI, physics, and cognitive science for applications like robotics and scene understanding. As this project was paused mid-progress due to project reprioritization, it includes incomplete sections (e.g., hardware considerations applied for causality) but demonstrates my ability to analyze complex research and propose innovative directions. This work captures the state of the field in 2020 and laid the foundation for my ongoing research in AI-driven simulation and physical reasoning.

Table of Contents

Abstract

Intuitive theories are equated to “common-sense reasoning” in order to capture the idea that the concepts, laws, and capabilities comprising an infant’s ability to stack blocks (Lerer, Gross, & Fergus, 2016) or understand that someone is reaching for an apple due to hunger are crucial building blocks for the architecture of human intelligence (Gerstenberg & Tenenbaum, 2016). Intuitive physics engines (IPEs) can be implemented using probabilistic, generative models to contextualize and solve psychophysics experiments in real-world scenarios through the application of intuitive physics theories (Battaglia et al., 2013) for physical scene understanding and human intent estimation tasks. Performance is evaluated by whether the IPE can successfully predict/mimic humans’ judgments on psychophysics experiments through metrics such as fall prediction accuracy (Lerer, Gross, & Fergus, 2016). The agent’s performance can be compared to ground-truth probabilistic physics simulations for a variety of different scenarios (Battaglia et al., 2013).

In addition to covering current state-of-the-art intuitive physics approaches, the report will cover general simulation tools (used in domains such as gaming, animation, and engineering) that navigate tradeoffs between accuracy and real-time decision-making capabilities, contra intuitive physics, for similar physical modeling applications. Hardware considerations for the experimental setup for both general simulation tools and intuitive physics approaches will be covered in the report as well.

Definitions

As we will explore in this report, intuitive physics refers to the common-sense reasoning humans (and some AI models) use to predict and understand physical interactions in the world. It involves concepts including object motion, collisions, forces, and stability, which are critical for

completing tasks such as predicting when a stack of blocks might fall or when a rolling ball will slow down for navigation and interactions in real-world environments. Unlike traditional physics simulations that rely on precise equations, intuitive physics models often use probabilistic, generative, or learning-based approaches to approximate human-like judgments about physical events.

Pipeline

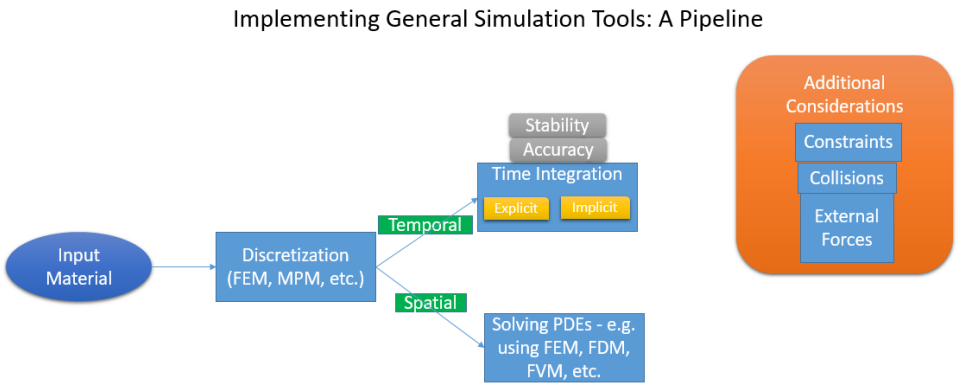

Figure 1: The pipeline used for general simulation tools. Source: Author.

The discretization of input material implemented by general simulation tools can be categorized into temporal and spatial discretization, which are summarized below:

Time Integration schemes are utilized to derive time-dependent world coordinates. Numerical solvers are used to solve a system of equations to compute the configurations for a discrete set of values of an unknown vector field at particular time steps - these vector fields are needed for the animation of the object. Explicit time integration schemes are defined as explicit formulas being present for quantities at the next time step; implicit time integration schemes are defined as implicit quantities being chosen as the solution for the system of equations. While explicit time integration schemes are convenient in terms of implementation, they can also be unstable and potentially result in an explosion of forces if

a large time step is chosen. Implicit time integration schemes provide stability against large time steps for stiff problems (e.g. materials strongly resisting deformation), but involve solving a “system of equations at each time step”. Nealen, Müller, Keiser, Boxerman, and Carlson (2005) note that stability and accuracy (in terms of convergence apropos time step size) are the two primary criteria for evaluating time integration schemes.

Spatial discretization involves “the discretization of spatially continuous quantities”, which should take into account choices on the usage of Eulerian vs. Lagrangian frames of reference, spatial data structures such as grids and meshes for storing simulation variables, and interpolation. Common methods for spatial discretization are finite differences methods or the finite element method.

Additional considerations include the applied constraints, how collision detection is implemented (the results vary apropos the form of the object, such as soft-body or rigid-body objects, and the algorithm of interest, among other factors), in addition to how external forces play a role in the simulation of objects.

Approaches

The characteristics of the approaches covered in this report are categorized into mesh-based and mesh-less methods, as depicted in Figure 2 and summarized in the following sections. Mesh-based methods involve the use of a mesh with information on nodal coordinates, connectivity, etc.; meshless methods avoid predefined relationships between nodes and offer flexibility for complex domains with higher computational cost compared to mesh-based methods (Trobec & Kosec, 2015).

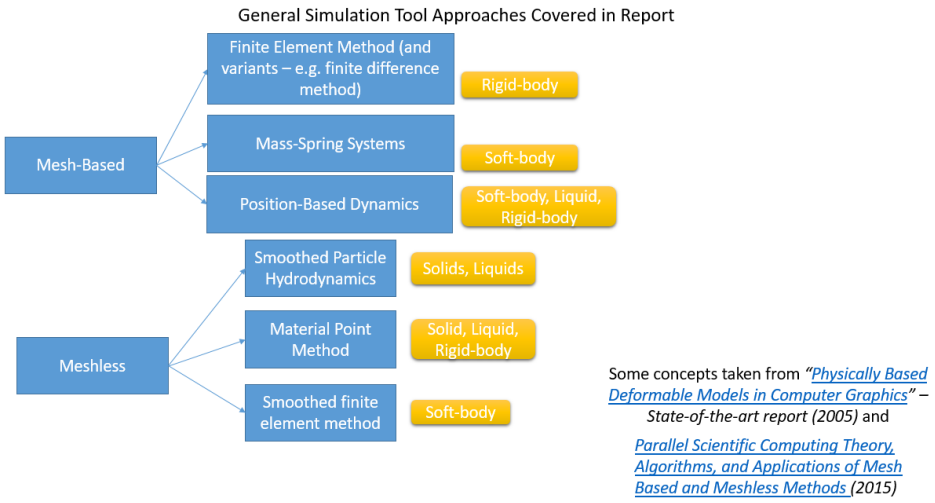

Figure 2: Characterization of simulation tool approaches covered in report into mesh-based and mesh-less domains

Mesh-Based

Finite Element Method and variants

The finite-element method involves discretizing a material into individual sub-domains/elements (in the form of a mesh) described locally by linear basis functions; adjacent elements will encompass common nodes, so "the mesh will define a piecewise function over the entire material" (O'Brien, & Hodgins, 1999, p. 3). Stated otherwise, the finite element method is leveraged to transform partial differential equations (PDEs) into algebraic equations that need to be solved numerically (Nealen, Müller, Keiser, Boxerman, & Carlson, 2005).

One variant of FEM is the finite differences method, where discretization of the equation of motion is done by utilizing finite differences, in addition to the Finite Volume method, where "the forces acting on the nodes of an element are computed as the derivatives of the deformation energy with respect to the nodal positions" (Nealen, Müller, Keiser, Boxerman, & Carlson, 2005, p. 6).

Position-Based Dynamics

Position-based dynamics is an approach that enables the ability to manipulate the positions of vertices and parts of objects in addition to control over the integration scheme. Objects are represented in terms of particles and constraints that are input to a symplectic Euler integration step that generates the new locations, which are used in addition to the original positions to implement the continuous collision detection. Following these steps, the predicted positions are iteratively corrected in relation to constraints through the solver, and the positions and velocities are then updated (Bender, Müller, Otaduy, & Teschner, 2013). In the case of research by Müller, Heidelberger, Hennix, and Ratcliff (2006), the integration scheme could not be defined as an explicit scheme or implicit scheme.

Details of the implementation of the PBD algorithm can be altered/extended for better computational cost, robustness, accuracy, etc. For example position-based approaches can be utilized for various applications using different types of meshes (air meshes lead to lower computational cost), lattices in relation to sampling approaches (e.g. regular lattices lead to lower computational cost for fine sampling),

Fratarcangeli and Pellacini (2013) present a disadvantage of the PBD algorithm – iteratively solving a large set of constraints can lead to increase in the computational cost for each iteration. Another disadvantage of PBD is lack of accuracy (in particular, when compared to force-based methods); for PBD, accuracy is considered to be less important compared to speed, stability, etc. and is measured in terms of visual plausibility. PBD's accuracy can be improved through modifications and extensions, such as adding orientation information to particles.

Mass-Spring Systems

Mass-spring systems involve a discrete model with point masses connected together by massless springs; forces on the masses are computed in relation to the mass' relationships/string connections with its neighbors (Nealen, Müller, Keiser, Boxerman, & Carlson, 2005). The accuracy of mass-spring systems is dependent on the hyperparameters of the algorithm, such as spring constants; Nealen, Müller, Keiser, Boxerman, and Carlson (2005) state that mass-spring systems are mostly not convergent and possess accuracy issues.

Meshless-Based

Smoothed particle hydrodynamics

Smoothed particle hydrodynamics (SPH) is a particle-based Lagrangian technique involving the use of the Monte Carlo approach to solve multi-dimensional hydrodynamic equations (Steinmetz, Matthias & Mueller, 1993); the particles are used to compute approximations of physical quantities and the associated spatial derivatives. A large number of particles can lead to higher accuracy, although a smaller number of particles can achieve high accuracy depending on the setup, extensions applied towards the SPH algorithm, etc. Nealen, Müller, Keiser, Boxerman, and Carlson (2005) state that a disadvantage of the SPH algorithm includes the inability to maintain the materials' incompressibility.

Utilizing parallel algorithms for shared memory (a capability commonly supported by modern machines) might be important for utilizing the SPH algorithm, as problems such as different parallel threads adding forces to the same particle can arise (Nishiura, Furuichi, & Sakaguchi, 2015). Furthermore, for SPH, the computational complexity for a system with N

particles is *O(N²), O(N logN)*, or *O(N)* depending on the algorithm used for the neighboring particle search (He, Huang, Liu, Li, Gan, & Sun, 2016).

Material Point Method

Adapting FEM for soft body and liquid simulation applications often involves the use of additional computation. The Material Point Method (MPM) involves the representation of a material as Lagrangian particles representing continuous pieces of material (where quantities such as density and force are continuous functions of position) (Jiang, Schroeder, Teran, Stomakhin, & Selle, 2016). Figure 3 provides an overview of the material point method.

He, Huang, Liu, Li, Gan, and Sun (2016) demonstrate that MPM has clear performance benefits in terms of memory usage and computational cost, as the MPM algorithm does not involve neighboring particle search.

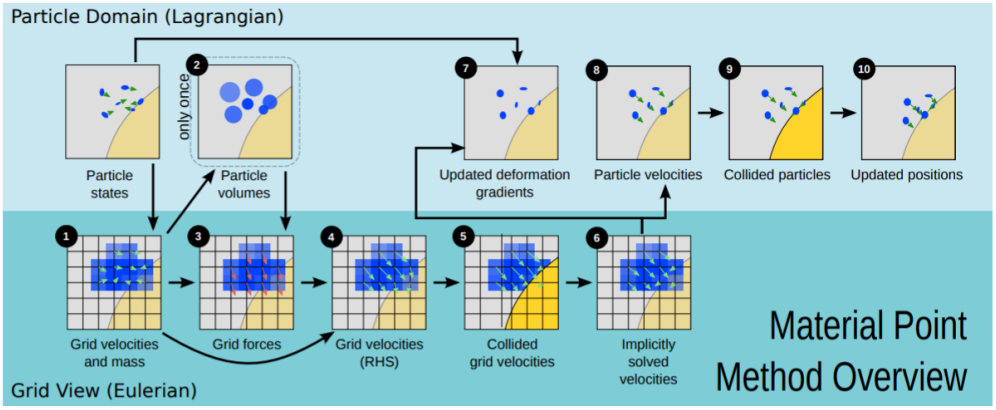

Figure 3: Material Point Method Overview. Source: (Jiang, Schroeder, Teran, Stomakhin, & Selle, 2016).

Smoothed finite element method

The smoothed finite-element method (S-FEM) is an improvement upon FEM, and involves using smoothing domains on top of the finite element mesh. Zeng and Liu (2016) notes S-FEM demonstrates higher accuracy, robustness, and convergence rates compared to FEM due to the "softening effect" the S-FEM approach provides in addition to the lack of a mapping procedure. However, while delivering these benefits, S-FEM has been demonstrated to have more computational cost compared to FEM, due to a larger number of nodes needed for the smoothing domain, which increases the bandwidth of the stiffness matrix of the model.

Method	Computational Cost	Memory Requirements	Complexity of Implementation	Accuracy
Material Point Method	**Low (compared to FEM); High (compared to SPH) (Depends on approximations of fields)**	**Medium – Lower compared to SPH (Depends on no. of particles being simulated, algorithm for transfer scheme, etc.)**	**High**	**Medium (Shape of assumed fields affects accuracy) – Higher than SPH**
Finite Element Method	**High**	**High**	**High**	**High**
Spring/Mass Models	**Low**	**Medium (No. of memory accesses and memory space depends on implementation, data structures used, etc.)**	**Low**	**Low (e.g. compared to FEM)**
Position-Based Dynamics	**Medium (depends on mesh, sampling approach, number of constraints, etc.)**	**Unknown – (depends on algorithm implementation and hardware used)**	**Low**	**Low (depending on approach taken)**

Smoothed Finite-Element Method	**High (compared to FEM)**	**Low**	**Low**	**High**
Smoothed Particle Hydrodynamics	**High**	**High**	**Low**	**High (with increasing number of particles)**

Sources: On the capabilities and limits of smoothed particle hydrodynamics, (Nealen, Müller, Keiser, Boxerman, & Carlson, 2005), (He, Huang, Liu, Li, Gan, & Sun, 2016), (Farmaga, Shmigelskyi, Spiewak, & Ciupinski, 2011), (Fratarcangeli & Pellacini, 2013), (Dong & Grabe, 2018), MPM or FEM / PFEM / SPH /... Insights on the Material Point Method future, Accuracy, errors and convergence of MPM, Improving the Material-Point Method, Lecture VII: Position-Based Dynamics, (Bender, Müller, Otaduy, & Teschner, 2013), A Survey on Position Based Dynamics, 2017

1. Rigid bodies

Rigid bodies are objects for which deformation is not relevant. In intuitive physics applications, rigid bodies can be represented with a single point mass in terms of the object's position and orientation in space, in addition to the object's mass, velocities, behavior when subject to friction and gravity, etc. (Battaglia, Hamrick, and Tenenbaum, 2013).

1.1 Intuitive physics

Various methodologies and topologies can be utilized for simulating physical reasoning for rigid objects, which are covered in this section.

Battaglia, Hamrick, and Tenenbaum (2013) approach simulation of rigid body dynamics through an IPE that utilizes Bayesian cognitive modeling, graphics, and simulation tools; represents mechanics procedurally with an emphasis on speed and generality; and takes into account uncertainty associated with the scene's state and external forces. The input to the authors' IPE model is a sample from a distribution over aspects of the 3D space including scene configurations, object properties, and external forces. The IPE model computes expected values of physical property queries (such as "Will the object fall?" or "In which direction?") to map an initial state and sequence of future states to judgements. Stochastic Monte Carlo simulations are leveraged via the Open Dynamics Engine (ODE), which represents the geometries of the objects as polyhedra and mass distributions as inertial tensors, to form approximate posterior distributions over future states. The model incorporates optimizations such as simplification of object properties and efficient encoding of probabilities by using one/few of the samples to mimic human behavior, in relation to adopting cheap approximations in simpler decision settings. Parameters that are controlled include uncertainty, magnitude of latent forces, and physical properties varying across objects (e.g. elasticity).

Sanborn and Mansinghka (2013) discuss the noisy Newton approach as a framework utilizing Bayes' rule that can be used to derive probabilistic inferences about an object generating sensory information, associated physical forces/interactions, and the state of the world. The noisy Newton model infers details about hidden variables, such as objects' masses and the coefficient of restitution (i.e. the ratio of the relative velocity following collision and the initial relative velocity), from *noisy* observations of final and initial velocities and prior beliefs using Bayesian inference. The authors determine that in experimental setups where motor object bias was present (involving tests for bias that initially moving objects are heavier), the noisy Newton model successfully predicted the bias and provided a good quantitative fit compared to other models that were evaluated. The authors note that it is possible to learn to make judgements about on object's mass based on probabilistic inferences by substituting prior experience for evaluation of the Newtonian mechanics' equations.

Wu, Yildirim, Lim, Freeman, and Tenenbaum (2015) present a generative model consisting of a) physical rigid body representations (encompassing mass, a friction coefficient, 3D shape, and position offset) fed as input to an IPE; b) an IPE that simulates the velocity profiles and positions for each object forward in time; and c) a likelihood function paired with a standard tracking algorithm that maps real-world videos to a velocity space in relation to the model's hypotheses of the velocity vectors. The output of the generative model is then fed to a deep convolutional network to enable predictions on objects' physical properties for outcome and mass prediction tasks.

Mrowca et al. (2018) discusses how rigid-bodies can be modeled using a hierarchical graph-based object representation, where an object is decomposed into a set of particles, where each particle's state of time is described with a vector consisting of the particle's position,

velocity, and mass. These particles are grouped into hierarchies depending on pairwise relations encoding material properties; for the purpose of rigid body simulations, pairwise relations help ensure the distance between particles is maintained. The authors implemented a hierarchical relation network that takes into account previous particle states, relations, and external effects from three separate modules (external forces, collisions, and history), and propagates these effects through the network's hierarchical graph convolutions using a propagation module. A state predictor module is then used to predict local future delta positions of particles based on the state of the particle, the total propagated effect on the particle, and gravity.

1.2 General Simulation Tools

Bender, Erleben, and Trinkle (2013) present a selection of general simulation solutions for interactive rigid body solutions, such as instantaneous-time dynamic models, used to capture the motions of rigid bodies over time. Discretizing this type of model allows the solution to form different complementarity problems in relation to a set of given constraints. Numerical methods, such as compute-intensive Direct Methods, computationally efficient iterative fixed-point schemes utilized by popular open-source simulators, and reduced coordinate formulation models (involving simulation of objects with a tree-structure) are then applied to compute interactive rigid body simulations.

Another popular solution utilized for rigid body simulation is FEM. Campos and Alfredo (2018) implement a nonlinear finite element solver, as an extension of FEM, to capture the distribution of mass of rigid bodies and interactions in a multibody environment. The authors utilize a single node ridge body element that take into account an object's inertia and mass properties (as a substitute for the traditional finite element meshes for the whole model), which are then connected to the rest of the model through a rigid constraint

formulation. The authors implement surface parameterizations as part of the contact formulation for rigid bodies.

Alternatively, Deul, Charrier, and Bender (2016) demonstrate a method of applying a PBD framework for solving joint and contact constraints of and between rigid bodies. The method involves modifications such as updating the velocities of constraints when correction displacement are applied for rigid bodies, considering the adjustment of hyper parameters for collision handling, and when to perform position effects and approximations in relation to the velocity level. Connectors, defined as points/vectors used for defining constraints that are located in a rigid body's local coordinates, were utilized to model the two-way coupling of rigid bodies and particles used to simulate deformable bodies.

1.3 Hardware Considerations

Bender, Erleben, and Trinkle (2013) discussed the possibility of building parallel solvers for solving a system of linear equations associated with simulating bilateral constraints to leverage parallelization for efficiency. The authors present optimized data structures for sparse matrix operations on the GPU, in addition to an efficient memory layout, as additional hardware optimizations. In addition, the authors proposed decomposition of rigid bodies into smaller parts as a potential solution for enabling simpler parallelization for collision detection.

As described above, Mrowca et al. (2018) utilize a hierarchical particle-based object representation similar to the recommendation posited by Bender, Erleben, and Trinkle (2013). Mrowca et al. (2018) utilized multiple Nvidia Titan Xp GPUs for the model training in addition to Unity3D with the FleX physics engine API to build an interactive particle-based

environment connected to a python interface, allowing the user to tweak free parameters and observe the effects on the object or individual particles. The authors note that increasing the number of particles leads to increased simulation time, which suggests support for numerous shapes/particle representations might vary for specialized accelerator configurations and require additional hardware optimizations.

2.0 Soft bodies

Soft body dynamics is a research field that involves simulations of deformable objects. Soft body simulations are expected to take into account maintenance of the shape, while considering the relative distance of two points is not fixed.

2.1 Intuitive physics

The research by Mrowca et al. (2018) on a Hierarchical Relation Network can also be used for soft bodies. For these types of objects, the relationships between particles represent local material stiffness, and imposing pairwise distance between the particles can be used to ensure the local pairwise deformations are correctly learned.

2.2 General Simulation Tools

Irving, Schroeder, and Fedkiw (2007) approach modelling of highly deformable solids using a fluid dynamics approach; the authors present a modelling technique that can be integrated into a finite element solver. A time discretization method is presented that takes into account correction of volume errors and position errors separately (without introducing oscillations). The authors' custom spatial discretization approach enforces volume preservation or compressibility for a composite element of a mesh consisting of

tetrahedrons. The authors define a set of (linear) constraints as part of the Poisson equations utilized by the time discretization method for object collisions and self-collisions.

Zhang et al. (2017) approach 3D object deformation simulation by applying S-FEM in addition to a stiffness warping approach (for extracting the rotational components from deformations). The simulation procedure involves computing and utilizing smoothed strain-displacement and smoothed stiffness matrices, consisting of parameters such as the number of nodes, external forces, etc. As part of the experimental setup, soft body objects were subjected to pressure, large rotations, rotational deformations, softening, etc. and effects on distorted meshes comprised of tetrahedron elements were measured.

Danevičius, Maskeliūnas, Damaševičius, Połap, and Woźniak (2018) discusses the application of spring/mass models for soft body simulation; this method involves linking particles to each other using a soft spring or elastic joint dependent on parameters such as the elasticity constant, damping constant, velocity of linked particles etc.

Stomakhin, Schroeder, Chai, Teran, and Selle (2013) present the use of MPM for simulating snow, which can be considered as soft-body material with both solid and fluid properties. The authors note that the benefits of MPM, such as tracking mass and other object properties through the Lagrangian particles, using the Cartesian grid as a scratch-pad for "avoiding high valence communication patterns from nearest-neighbor queries" (p. 4), the ability to handle plasticity well, etc. make it an efficient method for simulating different forms of snow varying in stiffness and other factors.

2.3 Hardware Considerations

Soft body simulations require more computations compared to rigid body simulations.

Irving, Schroeder, and Fedkiw (2007) utilize a 3GHz Xeon machine as part of the experimental setup; the authors note “the computational cost was 18 s/frame..., 25 s/frame with Poisson’s ratio .45, and 3.4 min/frame with Poisson’s ratio .499” (p. 5).

Danevičius, Maskeliūnas, Damaševičius, Połap, and Woźniak (2018) discuss the possibly of offloading soft-body physics computations to cloud. An Intelligent Offloading Management Module was implemented to determine compute-intensive computations to be run on an 4.3 GHz Intel i9-7900x CPU with relevant computations offloaded to the cloud. Figure 4 depicts the intelligent offloading management procedure utilized.

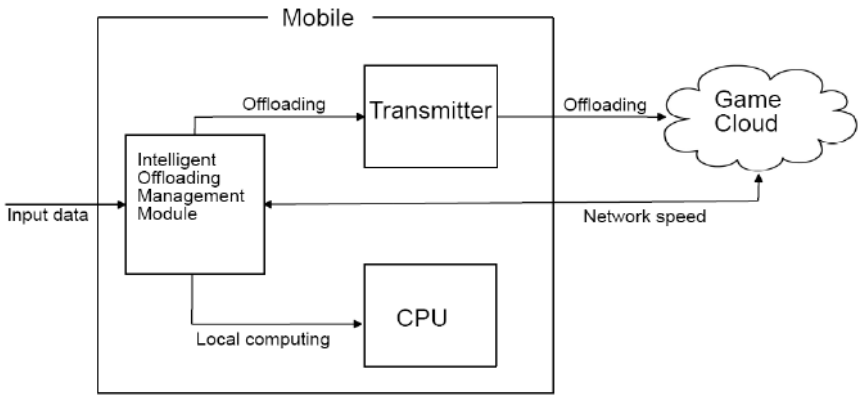

Figure 4: An intelligent offloading management procedure for soft body simulation (Danevičius, Maskeliūnas, Damaševičius, Połap, & Woźniak, 2018)

3.0 Liquids

The computations associated with liquid simulation are more complicated and compute-intensive compared to rigid body simulation, considering the number of degrees of freedom associated with fluids. As a way of comparison, real fluids can be represented on the order of 10^{23} particles per kilogram (Bates, Yildirim, Tenenbaum, and Battaglia, 2018).

3.1 Intuitive physics

Bates, Yildirim, Tenenbaum, and Battaglia (2015) evaluate the performance of two simulation-based models, a smoothed particle hydrodynamics (SPH) method and an intuitive fluids engine (IFE) that utilizes the particle-based simulation from SPH for predicting the movement of the liquid; more information on the authors' implementation of the SPH method and particle representation is present in the next section.

The intuitive fluids engine described in the paper takes as input the scene configuration (e.g. viscosity of liquid, presence of solid elements, particle friction, decreases in a particle's acceleration apropos the particles' velocity, particle "stickiness", etc.), the number of

particles to be instantiated, and physical uncertainty (implemented as a random offset represented by averaging predictions over a range of damping values). The IFE then runs a set of independent simulations, and forms a judgement based on a probability distribution over predictions of particles' future states.

Bates, Yildirim, Tenenbaum, and Battaglia (2018) tested the IFE discussed above in various scenarios for low and high viscosity liquids, and determined representational simplifications related to reducing the number of particles, simplifying particle-particle and particle-rigid interactions, etc. matches performance in relation to human judgments.

While not strictly considered under the intuitive physics domain, Schenck and Fox (2018) demonstrate the ability to leverage fully convolutional networks to solve psychophysics experiments related to perception and reasoning about liquids in real-time. The implementation involves the use of fully-connected neural network layers, long short-term memory (LSTM) layers to detect image pixels corresponding to liquid. The output of the network is a distribution over the liquid volume that is then fed as input to a hidden Markov model that models changes in the volume over time.

3.2 General Simulation Tools and Other Models

In the research by Bates, Yildirim, Tenenbaum, and Battaglia (2015), the state of the fluid is decomposed into a set of particles consisting of information in terms of the volume of fluid, position, velocity, pressure, etc. The SPH method involves interpolation of neighboring particles' properties, weighted by their distances to calculate density, pressure, and dynamics of a particle, which are consequently updated as part of the liquid simulation.

Researchers at Pixar constructed a particle-based multi-scale method utilizing SPH that can perform high resolution, incompressible fluid simulations and handle complex collision boundary conditions with reduced computation time and storage requirements (Horvath & Solenthaler, 2013). The traditional two-scale method utilized for SPH applications involves interpolations of scalars/vector fields stored at a particle from neighboring particles, and enables intrinsic conservation of mass. The method adopted by the authors builds upon the two-scale method for applications for higher resolution levels and is inspired by multi-scale models where resources are allocated to visually interesting regions. One relevant detail from the paper is the implementation of a particle emission strategy that tracks the total mass of objects and compensates for the volume loss, enabling computational efficiency.

3.3 Hardware considerations

Armfield, Morgan, and Srinivas (2002) suggest relevant recommendations for CFD methods include the use of parallel computers, “easy post processing for extremely large-scale computations”, etc. (p. 78).

Schenck and Fox (2018) state that “generating one 15 second sequence takes about 7.5 hours to simulate the liquid and an additional 0.5 hours to render it on our Intel Core i7 CPUs” (p. 4). The authors propose separating the rendering and liquid simulation components of the pipeline to reduce compute and memory requirements.

It is also relevant to consider hardware considerations and optimizations for general simulation tool approaches applied towards the modeling of liquid dynamics for insights that can be used to optimize intuitive physics pipelines, or pipelines consisting of both

intuitive physics engines and general simulation tools (for example, the research by Bates, Yildirim, Tenenbaum, and Battaglia (2015)).

Harada, Koshizuka, and Kawaguchi (2007) present the considerations and implementation details involved with deploying an SPH algorithm (where each time step is composed of bucket generation, density computation, a velocity update, and a position update) solely on a GPU, without having to offload the neighboring particle search computations associated with the method onto a CPU. The authors demonstrated the feasibility of accelerating the offline and real-time simulations of an SPH algorithm on a GPU, and showed that the computational speed achieved with this approach was approximately 28 times faster than implementing the algorithm on a CPU (shown in Figure 5).

Table 1 Times (a) and (b) are the computation times for bucket generation and computation time for one time step including the rendering time (in milliseconds).

Number of particles	Time (a)	Time (b)
1,024	0.75	3.9
4,096	0.80	5.45
16,386	1.55	14.8
65,536	3.99	58.6
262,144	14.8	235.9
1,048,576	55.4	1096.8
4,194,304	192.9	3783.6

Table 2 Computation time on CPUs (in milliseconds) and speed increase of the proposed method (times faster).

Number of particles	Time	Speed increase
1,024	15.6	4.0
4,096	43.6	8.0
16,386	206.2	13.9
65,536	1018.6	17.3
262,144	6725.6	28.5

Figure 5: Results on implementing SPM algorithm for GPU vs. CPU (Harada, Koshizuka, Kawaguchi, 2007)

The results from Figure 3 and the research suggest that GPUs might be better suited for the implementation of general simulation tools (and potentially IPEs that rely on them), in addition to how the number of particles quickly leads to larger values of time on the CPU compared to the implementation for GPU.

4.0 Causality

4.1 Intuitive physics

In previous sections, the concept of physical uncertainty that represents the effects of external uncertain factors and intuitive judgements on predictions of an object's trajectory or future state was discussed. Causal attributes can play a key role in human's intuitive understanding of physics for outcome simulations.

Causal reasoning extends the scope of inductive reasoning capabilities (i.e. inferring the cause for an effect) demonstrated by probabilistic graph models by taking into account the roles objects play in certain scenarios. For example, physics simulation models that take into account causal attributions can differentiate between an object "causing" an outcome (where a force was not directed or intended towards a certain state") and "helping" an outcome (where a force is directed towards a certain state) (Gerstenberg & Tenenbaum, 2016). Furthermore, in order to model humans' intuitive understanding of physics using causal judgements, counterfactuals (scenarios that almost happened or were prevented) are also an important aspect to consider.

4.2 Hardware Considerations

The hardware requirements for supporting causal attributions in physics simulation models are out of the scope of this report for now. (Note: This section was planned but not completed due to the project's halt in 2020).

5.0 Active (agents) and passive (objects)

This section will consider passive objects in relation to active agents capable of applying forces.

5.1 Intuitive physics

Wu, Yildirim, Lim, Freeman, and Tenenbaum (2015) implemented their algorithm for dynamic videos, where object A is on an inclined surface and can slide down and collide with object B; in this case, it could be said that object A is an active agent whose behavior is exhibited in dynamic scenes. The experimental setup for static scenes is similar, but the authors use the length of the video to govern the number of simulation steps used to output the velocity vectors for the given objects in addition to a tracking algorithm that initializes the objects' shapes, position offsets, and generates the velocity vectors used for determining the length of the generative model's output.

Bramley, Gerstenberg, Tenenbaum, and Gureckis (2018) state that active learning can be utilized to mimic causal Bayesian networks for modeling active interventions; for example, a user's mouse movement is not causally explainable but can affect the physical trajectories of objects. The authors use an Ideal-Observer (IO) approach in order to model both active (where the user exerts control) and passive learning (where the user watches sessions where previous users exerted control) scenarios; the IO approach is targeted at predicting different scene configurations in relation to observations in order to infer objects' latent properties.

An interesting use case is the scenario where the robot is directly involved in an active learning methodology, which is termed experiential learning. Agrawal, Nair, Abbeel, Malik, and Levine (2016) present the joint training of forward kinematics (defined as predicting

next state from the current state and action) and inverse kinematics models (defined as predicting the action from the initial state and the target state) as a solution for multi-step decision making and experiential learning.

6.0 Data-driven (neural representations)

The design of facets of intuitive physics engines is in part inspired by neural correlates of physics predictions systems. Schwettmann, Tenenbaum, and Kanwisher (2018) summarize the facets composing an intuitive physics model used for human physical reasoning as being “quantitative, approximate, compositional, and probabilistic” (p. 1).

Fischer, Mikhael, Tenenbaum, and Kanwisher (2016) demonstrated that brain activation patterns differ for different psychophysical experiments; these differences may be reflective of different aspects of physical scene understanding inferences. Furthermore, it seems that regions of the brain activated in response to intuitive physics tasks also serve as hubs for capabilities such as motor action planning and general-problem solving, suggesting that algorithmic correlates of neural mechanisms should consider interconnections between intuitive physics, action planning, and problem-solving domains.

This section will address important aspects of the design of intuitive physics engines driven by neural representations and a comparison between the algorithmic and biological correlates of intuitive physics.

6.1 Factorization and Compositionality

In the context of intuitive physics, compositionality can be considered as the combination of smaller building blocks for the formation of larger parts. The implementation of compositionality is tied to factorization, involving translating a scene into object-based representations. Chang, Ullman, Torralba, and Tenenbaum (2017) present the benefits of implementing a neural physics engine (NPE) that utilizes a compositional structure for translating object dynamics into pairwise interactions, which in turn allows the NPE to incorporate a causal structure of object dynamics. The benefits of this approach include the ability to flexibly adapt to different scene configurations, and transfer physical knowledge to numerous objects in the scene. The authors implement an NPE architecture that relies on reusable compositionality comprised by the separate encapsulation of each object and associated pairwise interactions, which outperforms a model with no pairwise interactions and an LSTM. The NPE architecture consists of an encoder with a pairwise layer, 25 hidden units, and a 5-layer feed forward network in addition to a decoder (which is a 5-layerd network) without a pairwise layer that is fed a concatenation of the summed pairwise encodings and the input state of the object of interest as input.

6.2 Top-down vs. Bottom-up approaches

Algorithms for physical reasoning can be categorized into two approaches – the top-down approach, involving inference over an IPE's parameters, providing generalization capabilities such as the work presented in this report by Battaglia et al. (2013) and the bottom-up approach involving mapping observations to physical judgements (avoiding pre-specification of certain details).

Oller, Wu, Wu, Tenenbaum, & Rodriguez (2019) define top-down learning as building abstractions and encoding macro-behaviors of the physical objects before considering learning motion models, and bottom-up learning as learning state-transition models using the abstractions and sensory data. The authors describe the abstractions as crucial for factoring the input space and for enabling the capability to identify moving/immobile objects. As part of their research, a generative probabilistic model is used to form a joint distribution over the states and abstractions, and MCMC sampling "with Hamiltonian dynamics to approximate the posterior distribution over states and abstractions given noisy measurements" (p. 5).

Chang, Ullman, Torralba, and Tenenbaum (2017) suggest a hybrid approach can help achieve the benefits of both approaches – their NPE builds upon structural assumptions, while adapting to object dynamics specific to scenes through joint training. Grzeszczuk, Terzopoulos, and Hinton (1998) in addition to Battaglia, Pascanu, Lai, Rezende, and Kavukcuoglu (2016) also took this approach with NeuroAnimator and interaction networks respectively.

6.2.1 Deep Neural Networks

One interesting detail unifying the implementation of these hybrid approaches is the use of deep neural networks or similar structures in the case of interaction networks for generalization/flexibility in addition to incorporating inductive biases.

While evidence for deep-learning models being effective for modeling human physical reasoning is still an outstanding question, approaches such as Galileo utilizing neural networks for learning from the experiences of the generative model and effectively mapping visual inputs to inferred attributes seem to match human performance relatively

well (Wu, Yildirim, Lim, Freeman, & Tenenbaum, 2015). Wu, Yildirim, Lim, Freeman, and Tenenbaum (2015) suggest that their implementation of a self-supervised learning algorithm being fed the outputs of the generative model as part of Galileo mimics Helmholtz machines and infants' cognitive development in terms of progressing from application of physical reasoning for dynamic videos to static scenes. An additional attribute that deep neural networks need to possess for these applications that probabilistic models might be better equipped for is the ability to handle perceptual uncertainty (noise due to initial measurements and imperfect velocities) and dynamics uncertainty (noise in the IPE), which are key components of human intuitive physical reasoning (Smith & Vul, 2012).

6.2.2 Unsupervised Learning

Ehrhardt, Monszpart, Mitra, & Vedaldi (2018) present an unsupervised learning methodology for physical reasoning applications that does not require the use of a simulator or training dataset. The proposed pipeline/methodology involves learning for object detection tasks implemented in an unsupervised manner, and the subsequently extracted positions to train a visual motion predictor. The unsupervised object tracker is pictured in Figure 6:

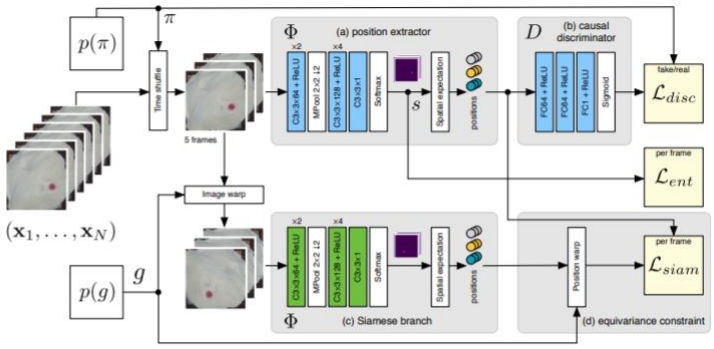

Fig. 1. Overview of our unsupervised object tracker Each training point consists of a sequence of five video frames. Top: the sequence is randomly permuted with probability 50%. The position extractor (a) computes a probability map s for the object location, whose entropy is penalised by $\mathcal{L}_{ent}$. The reconstructed trajectory is then fed to a causal/non-causal discriminator network (b) that determines whether the sequence is causal or not, encouraged by $\mathcal{L}_{disc}$. The bottom Siamese branch (c) of the architecture takes a randomly warped version of the video and is expected by $\mathcal{L}_{siam}$ to extract correspondingly-warped positions in (d). Blue and green blocks contain learnable weights and green blocks are siamese shared ones. At test time only Φ is retained.

Figure 6: Unsupervised Object Tracker. Source: Ehrhardt, Monszpart, Mitra, and Vedaldi (2018)

7.0 Recommendations

The research presented in this report suggests a topology incorporating a hierarchical probabilistic generative model (where the hierarchical aspects helps support

compositionality) and the usage of deep neural networks to perceive physical variables from scenes.

Furthermore, the research presented (e.g. Bates, Yildirim, Tenenbaum, and Battaglia (2015); Mrowca et al. (2018)) suggests that discretizing input objects using particle-based methods might lead to lower computational cost, better memory efficiency, lower complexity of implementation, and higher accuracy compared to other methods. Furthermore, the research by Bates, Yildirim, Tenenbaum, & Battaglia (2015) suggests that it might be possible to supplement the IPE with a general simulation tool, such as SPH.

One remaining question from this research would be whether an IPE consisting solely of a deep neural network can successfully incorporate key capabilities that probabilistic generative models offer for intuitive physics contexts such as compositionality, handling uncertainty, and generalizing and transferring knowledge for quick real-time decision-making capabilities. The research by Chang, Ullman, Torralba, and Tenenbaum (2017) and Grzeszczuk, Terzopoulos, and Hinton (1998) also inspires an additional question: Are deep neural networks or probabilistic generative models more efficient and closer to neural mechanisms when combining top-down and bottom-up approaches for physical reasoning?

The summary of hardware performance metrics for the different intuitive physics approaches and general simulation tools suggests implementing the algorithms on a GPU, with parallelization, in addition to specialized/optimized data structures and memory layout, can help achieve optimal results with low computational cost.

References

Agrawal, P., Nair, A., Abbeel, P., Malik, J., & Levine, S. (2016). Learning to Poke by Poking: Experiential Learning of Intuitive Physics. *CoRR, abs/1606.07419*.

Armfield, S., Morgan, P., & Srinivas, K. (2002). Computational Fluid Dynamics, presented at Proceedings of the Second International Conference on Computational Fluid Dynamics, ICCFD, Sydney, Australia, 2002. Berlin: Springer.

Bates, C., Battaglia, P.W., Yildirim, I., & Tenenbaum, J.B. (2015). Humans predict liquid dynamics using probabilistic simulation. *CogSci*.

Bates, C., Yildirim, I., Tenenbaum, J.B., & Battaglia, P.W. (2018). Modeling human intuitions about liquid flow with particle-based simulation. *CoRR, abs/1809.01524*.

Battaglia, P. W., Hamrick, J. B., & Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. Proceedings of the National Academy of Sciences of the United States of America, 110(45), 18327-32.

Battaglia, P.W., Pascanu, R., Lai, M., Rezende, D.J., & Kavukcuoglu, K. (2016). Interaction Networks for Learning about Objects, Relations and Physics. *NIPS*.

Bargteil, A.W., & Shinar, T. (2018). An introduction to physics-based animation. *SIGGRAPH Courses*.

Bender, J., Müller, M., Otaduy, M.A., & Teschner, M. (2013). Position-based Methods for the Simulation of Solid Objects in Computer Graphics. *Eurographics*.

Bramley, N.R., Gerstenberg, T., Tenenbaum, J.B., & Gureckis, T.M. (2018). Intuitive experimentation in the physical world. *Cognitive Psychology, 105*, 9-38.

Campos, P. R. R. d., & Alfredo, G.N., (2018). Rigid body formulation in a finite element context with contact interaction. *Computational Mechanics 62*(6), p. 1369-1398.

Chang, M., Ullman, T., Torralba, A., & Tenenbaum, J.B. (2017). A Compositional Object-Based Approach to Learning Physical Dynamics. *CoRR, abs/1612.00341*.

Danevičius, E., Maskeliūnas, R., Damaševičius, R., Połap, D., & Woźniak, M. (2018). A Soft Body Physics Simulator with Computational Offloading to the Cloud. *Information,9*(12), 318. doi:10.3390/info9120318

Dong, Y., & Grabe, J. (2018). Large scale parallelisation of the material point method with multiple GPUs. *Computers and Geotechnics, 101*, 149-158. doi:10.1016/j.compgeo.2018.04.001

Deul, C., Charrier, P., & Bender, J. (2016). Position-based rigid-body dynamics. *Journal of Visualization and Computer Animation, 27*, 103-112.

Ehrhardt, S., Monszpart, A., Mitra, N.J., & Vedaldi, A. (2018). Unsupervised Intuitive Physics from Visual Observations. *ACCV*.

Farmaga, I., Shmigelskyi, P., Spiewak, P., & Ciupinski, L. (2011). Evaluation of computational complexity of finite element analysis. *2011 11th International*

Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 213-214.

Fischer, J., Mikhael, J.G., Tenenbaum, J.B., & Kanwisher, N. (2016). Functional neuroanatomy of intuitive physical inference. *Proceedings of the National Academy of Sciences of the United States of America, 113 34*, E5072-81 .

Fratarcangeli, M., & Pellacini, F. (2013). A GPU-Based Implementation of Position Based Dynamics for Interactive Deformable Bodies. *J. Graphics Tools, 17*, 59-66.

Gao, M., Wang, X., Wu, K., Pradhana, A., Sifakis, E., Yuksel, C., & Jiang, C. (2018). GPU optimization of material point methods. *ACM Trans. Graph., 37*, 254:1-254:12.

Gerstenberg, T., & Tenenbaum, J. B. (2016). Intuitive Theories. 1-50. Retrieved from https://cbmm.mit.edu/sites/default/files/publications/Intuitive Theories (Gerstenberg, Tenenbaum, 2016.pdf

Grzeszczuk, R., Terzopoulos, D., & Hinton, G.E. (1998). NeuroAnimator: Fast Neural Network Emulation and Control of Physics-based Models. *SIGGRAPH*.

Harada, T., Koshizuka, S., & Kawaguchi, Y. (2007). Smoothed Particle Hydrodynamics on GPUs. *Computer Graphics International.*

He, L., Huang, Z., Liu, H., Li, H., Gan, Y., & Sun, Z. (2016). Material point method and smoothed particle hydrodynamics simulations of fluid flow problems: A comparative

study. *Progress in Computational Fluid Dynamics, An International Journal, 1*(1), 1. doi:10.1504/pcfd.2016.10001222

Horvath, C. J., & Solenthaler, B. (2013). Mass Preserving Multi-Scale SPH: Pixar Technical Memo #13-04.

Irving, G., Schroeder, C., & Fedkiw, R. (2007). Volume conserving finite element simulations of deformable models. *SIGGRAPH '07*.

Jiang, C., Schroeder, C.A., Teran, J., Stomakhin, A., & Selle, A. (2016). The material point method for simulating continuum materials. *SIGGRAPH Courses*.

Lerer, A., Gross, S., & Fergus, R. (2016). Learning Physical Intuition of Block Towers by Example. ICML.

Müller, M., Heidelberger, B., Hennix, M., & Ratcliff, J. (2006). Position Based Dynamics. *VRIPHYS*.

Mrowca, D., Zhuang, C., Wang, E., Haber, N., Fei-Fei, L., Tenenbaum, J.B., & Yamins, D.L. (2018). Flexible neural representation for physics prediction. *NeurIPS*.

Nealen, A., Müller, M., Keiser, R., Boxerman, E., & Carlson, M. (2005). Physically Based Deformable Models in Computer Graphics. *Eurographics*.

Nishiura, D., Furuichi, M., & Sakaguchi, H. (2015). Computational performance of a smoothed particle hydrodynamics simulation for shared-memory parallel

computing. *Computer Physics Communications,194*, 18-32. doi:10.1016/j.cpc.2015.04.006

O'Brien, J.F., & Hodgins, J.K. (1999). Graphical Modeling and Animation of Brittle Fracture. *SIGGRAPH*.

Oller, M., Wu, J., Wu, Z., Tenenbaum, J.B., & Rodriguez, A. (2019). See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion. *Science Robotics, 4*.

Sanborn, A., Mansinghka, V.K., & Griffiths, T.L. (2013). Reconciling intuitive physics and Newtonian mechanics for colliding objects. *Psychological review, 120 2*, 411-37.

Schenck, C., & Fox, D. (2018). Perceiving and Reasoning About Liquids Using Fully Convolutional Networks. *I. J. Robotics Res., 37*, 452-471.

Schwettmann, S., Fischer, J., Tenenbaum, J.B., & Kanwisher, N. (2018). Evidence for an Intuitive Physics Engine in the Human Brain. *CogSci*.

Smith, K.A., & Vul, E. (2012). Sources of Uncertainty in Intuitive Physics. *Topics in cognitive science, 5 1*, 185-99 .

Steinmetz, Matthias & Mueller, E. (1993). On the capabilities and limits of smoothed particle hydrodynamics. *Astronomy and Astrophysics. 268.* 391-410.

Stomakhin, A., Schroeder, C.A., Chai, L., Teran, J., & Selle, A. (2013). A material point method for snow simulation. *ACM Trans. Graph., 32*, 102:1-102:10.

Trobec, R. and Kosec, G. (2015). Parallel Scientific Computing. Theory, Algorithms, and Applications of Mesh Based and Meshless Methods. *Springer Intl. Publ.* https://doi.org/10.1007/978-3-319-17073-2

Tu, Jiyuan. & Inthavong, Kiao. & Ahmadi, Goodarz. (2013). *Computational fluid and particle dynamics in the human respiratory system*. Dordrecht: Springer

Wu, J., Yildirim, I., Lim, J. J., Freeman, B., & Tenenbaum, J. (2015). Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28* (pp. 127–135). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5780-galileo-perceiving-physical-object-properties-by-integrating-a-physics-engine-with-deep-learning.pdf

Zeng, W. and Liu, G. R. (2016) "Smoothed finite element methods (S-FEM): An overview and recent developments". *Arch. Comput. Method Eng,* 1–39.

Zhang, J., Zhou, M., Huang, Y., Ren, P., ZhongkeWu, XuesongWang, & Zhao, S. (2017). A Smoothed Finite Element-Based Elasticity Model for Soft Bodies.