Neural Re-Simulation for Generating Bounces in Single Images

Carlo Innamorati[1*], Bryan Russell[2], Danny M. Kaufman[2], and Niloy J. Mitra[1,2]

[1]University College London
[2]Adobe Research
http://geometry.cs.ucl.ac.uk/projects/2019/bounce-neural-resim/

Abstract

We introduce a method to generate videos of dynamic virtual objects plausibly interacting via collisions with a still image's environment. Given a starting trajectory, physically simulated with the estimated geometry of a single, static input image, we learn to 'correct' this trajectory to a visually plausible one via a neural network. The neural network can then be seen as learning to 'correct' traditional simulation output, generated with incomplete and imprecise world information, to obtain context-specific, visually plausible re-simulated output – a process we call neural re-simulation. We train our system on a set of 50k synthetic scenes where a virtual moving object (ball) has been physically simulated. We demonstrate our approach on both our synthetic dataset and a collection of real-life images depicting everyday scenes, obtaining consistent improvement over baseline alternatives throughout.

1. Introduction

> Christopher Robin: You're next, Tigger. Jump!
> Tigger: Er, jump? Tiggers don't jump, they bounce.
> Winnie the Pooh: Then bounce down.
> Tigger: Don't be "ridick-orous". Tiggers only bounce up!
> – A. A. Milne, Winnie The Pooh

A single still image depicts an instant in time. Videos, on the other hand, have the capacity to depict dynamic events where scene objects may interact with and bounce off each other over time. We seek to allow users to bring single still images to life by allowing them to automatically generate videos depicting a virtual object interacting with a depicted scene in the image.

Specifically, we address the problem of dynamic object compositing in a single still image where a virtual object physically interacts by contact, such as bouncing, with the depicted scene. Our goal is to generate *visually plausible* virtual object-scene interactions instead of a physically accurate forward prediction. In other words, we seek to generate an output that looks physically valid to a human observer even if it does not exactly match an observed physical interaction starting from the same initial conditions. Our task is important for applications in augmented reality and animation, allowing users to author dynamic events in a depicted scene without access to sufficient information about the corresponding world.

For our study, we focus on the case where we toss a virtual ball into an everyday scene, such as living rooms, bedrooms, and kitchens, and seek to have the ball bounce off different objects in the scene (see Figure 1). This setup presents many challenges as we need to reason about the geometric layout of the scene and infer how the virtual ball will physically interact with the scene. Previous approaches either do not physically interact with the scene through contact [12, 20], require captured scene depth through multiple views or active sensing [33, 34], or rely on forward simulation using predicted single-view scene depth. However, predicted scene depth may be noisy, resulting in inaccurate and visually implausible forward-simulated trajectory output (see videos in supplemental). Furthermore, there may be global scaling issues with depth predictions if an algorithm is trained on one dataset and applied to another with different camera intrinsics.

To address the above challenges, we seek to learn to 're-simulate' the outputs obtained from running a forward simulation over noisy, estimated scene geometry inferred for the input image to yield a visually plausible one. Resimulation methods are traditionally applied in visual effects to reuse and combine pre-exisiting physical simulations for new and novel scenarios [22, 39, 45]. Here, in analogy, we introduce a neural network for learning re-

[*] Work initiated at Adobe during CI's summer internship.

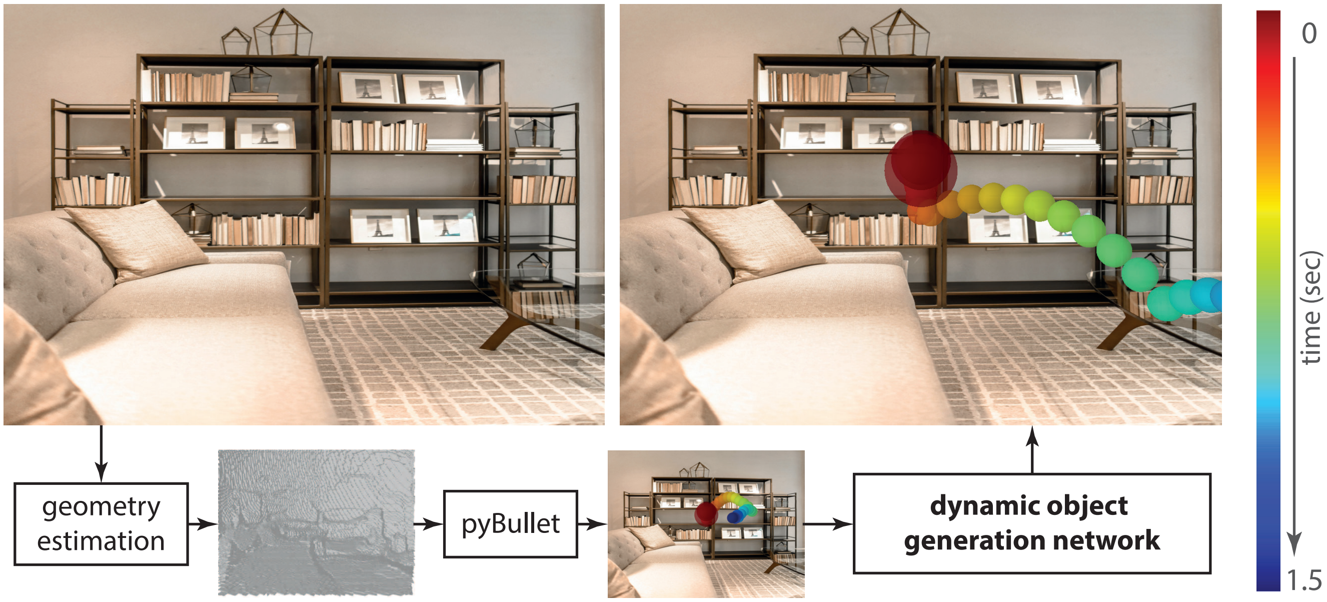

Figure 1. **Problem statement and approach overview.** We take as input a single still image depicting a scene and output a video depicting a virtual object dynamically interacting with the scene through bouncing. Here, we consider a ball as our virtual object. We achieve this by our Dynamic Object Generation Network which takes as inputs estimated depth and an initial forward trajectory of the virtual object from the PyBullet simulator [11] and outputs a 'corrected' trajectory via a neural re-simulation step. To visualize all the trajectories in this paper, we composited the virtual object at each time step with the input image; warmer colors indicate earlier time steps. **Please view output videos in the supplemental.**

simulation – a process we call *neural re-simulation*. In particular, here we apply neural re-simulation from trajectories with noisy and insufficient data to plausible output. Our solution is thus also related to recent approaches for re-rendering scenes with a neural network [21, 27, 29, 44]; here we seek to re-simulate dynamic trajectory outputs.

For the initial forward trajectory, we generate geometry based on estimated depth from the input image and run a physical simulator on this estimated geometry. Then, in the neural re-simulation step, our proposed model 'corrects' the initial trajectory resulting from the noisy depth predictions conditioned on context information available in the input image. Furthermore, we introduce an approach that learns to correct the global scaling of the estimated depths for the scene conditioned on the initial trajectory. We train our system on a dataset of trajectories computed by forward simulating trajectories with an off-the-shelf physical simulator (PyBullet [11]) on SUNCG scenes [42]. We find that our forward trajectories complement the information provided by the estimated depths. Finally, we use an adversarial loss [18] during training to allow for learning to generate visually plausible trajectory outputs.

Evaluating the quality of the dynamic object insertion task is difficult due to two factors: first, on real images, there is no available ground truth to compare against; and second, for target applications such as AR/VR and animation, 'visual plausibility' is more relevant rather than accurate forward trajectory predictions. We evaluate our proposed approach quantitatively both on synthetic data where we have access to ground truth simulations and on real data via a user study.

Our contribution is a system that learns to correct an initial trajectory of a virtual object provided by forward simulation on geometry specified by predicted depth of an input single still image to output a visually plausible trajectory of the object. Furthermore, we introduce a network that learns to update the predicted depth values conditioned on the initial trajectory. We demonstrate our approach on synthetic images from SUNCG and on real images and show that our approach consistently outperforms baseline methods by a healthy margin.

2. Related Work

Our work is primarily related to previous approaches on generating video and modeling dynamic object interactions.
Video generation. Prior work has generated dynamic or video textures by analyzing low-level motion features [13, 40]. However, we seek to model and generate motions due to high-level interactions in a scene. Prior work has looked at modeling interactions in constrained environments, such as sports [3]. While these interactions are complex and involve reasoning about the intention of multiple agents in a scene, we seek to model interactions that occur in everyday scenes. More recently, there has been work to endow

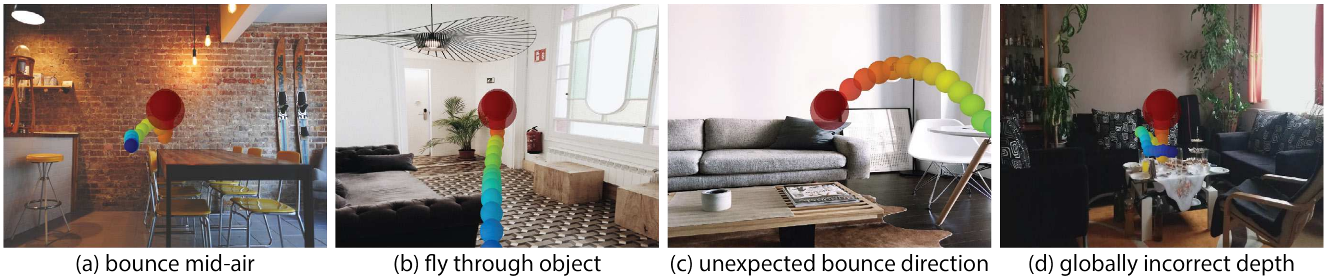

Figure 2. **Visually implausible trajectories.** Examples of the visually implausible trajectories that are generated by simulations with depth prediction. Left to right: a virtual object bounces in mid air, flies into an object, bounce in an unexpected direction, or has completely different scale due to globally incorrect depth.

a neural network with the ability to generate video. Examples include forecasting human dynamics from a single image [10], forecasting with variational auto-encoders [48], generating visual dynamics [53], and generating the future [47]. These works primarily forecast or generate human actions in video and do not focus on modeling object interactions. Moreover, making long-term video generations spanning multiple seconds from a neural network is challenging. Most relevant to our generation task is prior work that disentangles underlying structure and from the generation step [46].

Object interactions and intuitive physics. Prior work in modeling dynamic object interactions have involved reasoning and recovering parameters to a physical system [4, 7, 5, 6, 23, 26, 30, 55, 56]. While these works aim for physical accuracy, our aim is different as we want to achieve visual plausibility through learning. Other work has looked at changing the viewpoint of an object in a scene [20] or manipulating modal bases of an object [12], but do not address the problem of object-scene interaction via contact. Recent work has aimed to train a learning system to reason about physics for understanding the semantics of a scene [19] or for making future predictions in synthetic 2D scenarios [2, 9, 16, 50]. Work has aimed to go beyond the synthetic setting by inferring forces in real-world images through Newtonian image understanding [31]. More recently, work has aimed to learn to model 3D systems, such as predicting where toy blocks in a tower will fall [24], modeling a variety of different closed-world systems such as ramps, springs and bounces [14, 15, 51, 52], and predicting the effect of forces in images by reasoning over the 3D scene layout and training over 3D scene CAD models [32]. Recent work has also leveraged learning about real-world interactions by interacting with the world through visual-motor policies [25], grasping [37], pushing and poking objects [1, 36], crashing a drone into scene surfaces [17], hitting surfaces with a stick to reason about sound [35], or generating audible shapes [54]. Closest to ours is recent work that uses a neural network to make forward predictions of bounces in real-world scenes [38] and infer scene geometry and physical parameters such as coefficient of restitution [49]. Note that this work, while similar to ours, aims for physical accuracy and predicts an immediate short-term trajectory of a ball bounce after making contact with a surface. We go beyond this work and infer roll outs over multiple bounces and aim for visual plausibility.

3. Dynamic Object Generation Network

Given an input image $\mathcal{I}$ depicting a scene, we seek to output a video $\mathcal{V}$ of time length T where a virtual moving object with initial conditions ρ has been composited into the image depicting physical interaction with the scene. Example scene interactions include the virtual object flying in the air, making contact with several scene objects, and changing its trajectory after bouncing off scene objects. While one could learn to generate the video directly from training data using a neural network, such long-term generations are currently hard due to fundamental issues such as temporal flickering and decaying visual signal to the mean [28]. Recent work in long-term video generation has shown success by disentangling the prediction of the underlying scene structure and generation of the video pixels [46]. We seek to leverage this insight for our video generation task.

Our approach starts by computing an initial depth map Z_0 of the input image, which is then passed, after geometry processing, to a physical simulator $\mathcal{S}$ along with initial conditions ρ. The initial conditions consist of the initial state of the virtual object (velocity and position) and material parameters (*e.g.*, coefficients for friction and restitution). The output of the physical simulator is an initial trajectory $X_0 = \mathcal{S}(Z_0, \rho)$ represented as a sequence of object states $X_0 = (x_1, \dots, x_T)$. We assume access to depth predictions for the scene from an off-the-shelf algorithm. For our work, we use the depth prediction system of Chakrabarti *et al.* [8], which is a top performer on predicting depth on the NYUv2 dataset [41]. We run forward simulations using the PyBullet physical simulator [11].

Given the initial forward trajectory X_0, one could simply generate the final video $\mathcal{V}$ by passing the trajectory and the image $\mathcal{I}$ to a composing function $\mathcal{R}$ to yield $\mathcal{V} = \mathcal{R}(\mathcal{I}, X_0)$.

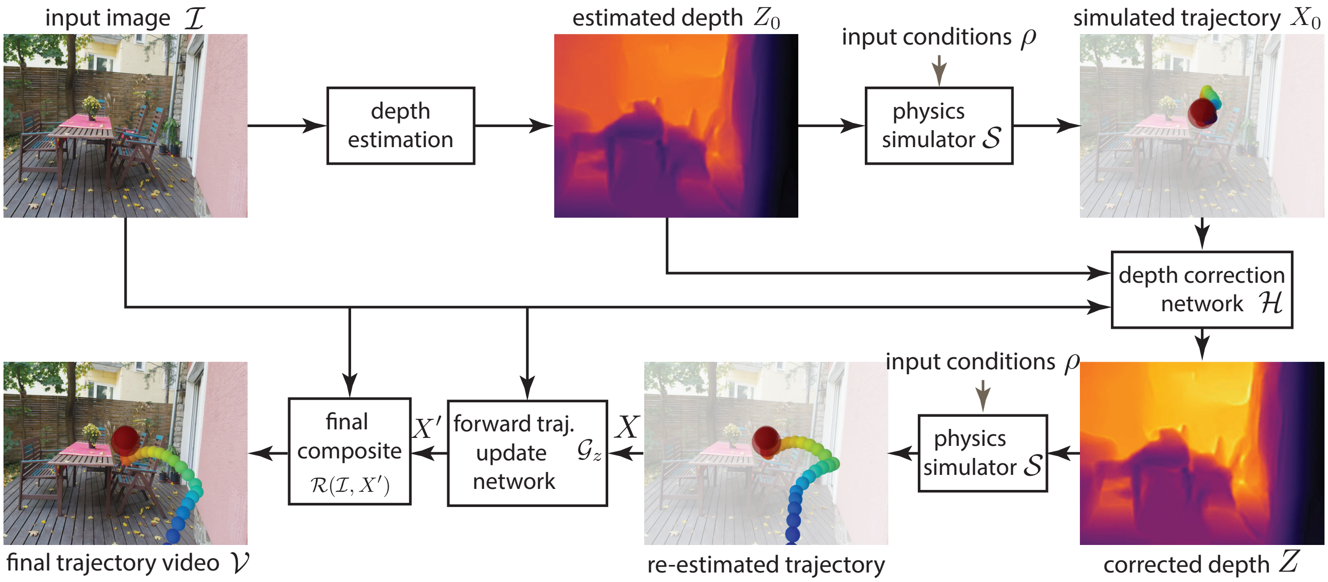
input image $\mathcal{I}$
depth estimation
estimated depth Z_0
input conditions ρ
physics simulator $\mathcal{S}$
simulated trajectory X_0
depth correction network $\mathcal{H}$
input conditions ρ
physics simulator $\mathcal{S}$
corrected depth Z
re-estimated trajectory
X
forward traj. update network $\mathcal{G}_z$
X'
final composite $\mathcal{R}(\mathcal{I}, X')$
final trajectory video $\mathcal{V}$

Figure 3. **System overview.** Our system takes as inputs an image depicting a scene and initial conditions for the object that is tossed in the scene and outputs a video showing a visually plausible predicted trajectory of the object interacting with the scene. Our approach predicts depth at every pixel in the image and consists of two networks – a forward trajectory update network $\mathcal{G}_z$ and a depth correction network $\mathcal{H}$. See text for more details.

However, this step often yields visually implausible output videos where the virtual object may or may not change direction at inappropriate times or bounce in a direction that is not congruent with a depicted surface due to inaccurate depth predictions. Example visually implausible outputs are shown in Figure 2. Such visually implausible artifacts cause the viewer to perceive the virtual object to bounce in mid-air, fly into a scene object or surface, or bounce in an unexpected direction, and are due to inaccurate predictions in the depth estimates.

Our insight is to correct such visually implausible artifacts by correcting the initial depth Z_0 and initial forward trajectory X_0 to yield a visually plausible final result – we call this step neural re-simulation. We introduce two networks $\mathcal{G}_z$ and $\mathcal{H}$ that generate updated forward trajectory X and updated depth map Z, respectively. The forward trajectory update network $\mathcal{G}_z$ is a generative neural network parameterized by scalar z that takes as inputs the image $\mathcal{I}$ and a forward trajectory X and returns an updated trajectory,

$$X' = \mathcal{G}_z(\mathcal{I}, X). \tag{1}$$

The depth correction network $\mathcal{H}$ is a neural network that returns an updated depth map given the image $\mathcal{I}$, initial depth map Z_0, and initial trajectory X_0 as inputs,

$$Z = \mathcal{H}(\mathcal{I}, Z_0, X_0). \tag{2}$$

The final video can be generated by composing the two networks,

$$\mathcal{V} = \mathcal{R}(\mathcal{I}, \mathcal{G}_z(\mathcal{I}, \mathcal{S}(\mathcal{H}(\mathcal{I}, Z_0, X_0), \rho))). \tag{3}$$

We describe both networks in the following subsections and outline our overall approach in Figure 3.

3.1. Trajectory update network

We assume a neural network for the trajectory update network. The network takes as inputs the input image $\mathcal{I}$, the virtual object's forward trajectory X resulting from the corrected depth predictions for the scene, and a value z sampled from a Gaussian distribution $z \sim \mathcal{N}(0, 1)$. The network first consists of a multilayer perceptron (MLP) that takes as input a concatenation of the trajectory X and sampled value z and outputs an encoded representation of the trajectory. The MLP is followed by a second MLP that takes as input a concatenation of the encoded trajectory representation and an encoding of the image $\mathcal{I}$. We obtain the image encoding through a pre-trained Inception-ResNet model [43] that has been pre-trained on ImageNet. The encoding is extracted from the penultimate layer. The second MLP outputs the updated trajectory X'.

Learning. One could train our trajectory update network using an L_2 loss to ground truth trajectories with the aim of making physically accurate predictions. However, this strategy would yield over-smooth predictions to the mean distribution over trajectories. Moreover, our goal is to generate visually plausible trajectories and not necessarily physically accurate ones.

To achieve our goal, we aim to fool a discriminator given a dataset of trajectories and the set of initial trajectories provided by running a forward simulation on the geometry from the initial depth map Z_0. We train the trajectory up-

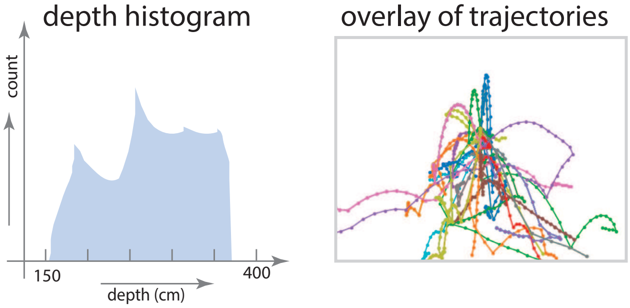

Figure 4. **Dataset.** Histogram of scene depth (left) and sampled trajectories from our dataset (right), illustrating the dataset's variety over depth and trajectory.

date network using an adversarial loss [18]. Given training examples of visually plausible trajectories $p_{plausible}$ and a set of initial trajectories $p_{initial}$, we seek to optimize the following adversarial loss over the trajectory update network $\mathcal{G}$ and a discriminator architecture $\mathcal{D}$,

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{X \sim p_{plausible}}[\log(\mathcal{D}(\mathcal{I}, X))] + \mathbb{E}_{\substack{X \sim p_{initial} \\ z \sim \mathcal{N}(0,1)}}[\log(1 - \mathcal{D}(\mathcal{I}, \mathcal{G}_z(\mathcal{I}, X)))]. \quad (4)$$

The discriminator network consists of an MLP that takes as input a trajectory X and outputs its encoded representation. The MLP is followed by a second MLP that takes as input a concatenation of the encoded trajectory representation and an Inception-ResNet encoding of the image $\mathcal{I}$. The second MLP outputs a prediction label.

To help with the early stages of training, the adversarial loss is aided by an L2 loss of decreasing relevance over training epochs. In particular, the L2 loss is weighted down by 0.5% after every epoch, resulting in a complete adversarial loss after the 200th epoch. The network is run for a total 1k epochs.

3.2. Depth correction network

A major source of error is when an initial depth map Z_0 is grossly out of range of the expected depth values for a given depicted scene. To correct this issue, we seek to have a network learn to output calibration parameters Z_{min} and Z_{max} that will be used to scale the initial depth values into the expected range. To achieve this result, we assume the depth correction network $\mathcal{H}$ is a MLP followed by a depth calibration update that takes as input a concatenated vector consisting of an encoded representation of the input image $\mathcal{I}$, an encoding of the initial depth map Z_0, and the output of the trajectory update network with the initial trajectory X_0 passed as input. The MLP outputs (Z_{min}, Z_{max}), which is then used to scale the min and max input depth map values in Z_0 to match the newly computed normalization values. The updated depth map Z is returned as output.

Learning. The depth correction network is trained using L_2 loss to regress the two normalization parameters given the trajectory X_0, an encoding of the image $\mathcal{I}$ and an encoding of the initial depth map Z_0. The network is run for a total of 1k epochs.

Geometry processing. To obtain the input trajectories of the model X_0 from the input depth map, we leverage projection and view matrices to obtain a point cloud, where each vertex corresponds to a pixel of the depth map. We then turn the point cloud into a mesh by connecting neighboring vertices. The mesh is then passed through PyBullet to obtain a corresponding trajectory.

4. Synthetic Trajectory Generation

A critical aspect of training our system is the ability to learn from a large collection of examples depicting an object interacting with an everyday scene. This aspect is challenging as such data is relatively scarce. For example, while one could consider real videos, the largest known dataset of a ball interacting with a scene contains about 5000 videos [38]. Moreover, the ball starts with different initial velocity (speed and direction) in each video.

To overcome this challenge, we leverage recent datasets containing large stores of 3D CAD models. We consider the SUNCG dataset for our study [42]. The SUNCG dataset contains 45k 3D scenes. We equip the 3D CAD models with a physics simulator, namely PyBullet [11]. To render forward trajectories, we import a scene into PyBullet and specify a camera viewpoint. We leverage the pre-computed camera locations provided by the SUNCG toolbox, which depicts viewpoints where the camera is held upright with the pitch angle rotated up by 30 degrees and pointing toward the interior of the scene geometry. We filtered the cameras to not directly face walls and other large surfaces and adjusted the cameras to match PyBullet's intrinsic parameters. We use 50k of the available 828k cameras.

For our study, we assume that the object is a spherical ball and starts with an initial velocity of 0.6 meters per second in the direction away from the camera center. We set the coefficients for friction and restitution to 0.5 and generate forward trajectories of length 1.5 seconds, sampled at 20Hz. At each time point, we record the output of the 3D center-of-mass location of the ball in camera space from PyBullet. After processing the trajectory, we render the new frames by re-creating the object in PyBullet and re-updating the coordinate system.

Our dataset consists of 50k examples, with each example coming from a multi-room 3D model containing on average eight different rooms. We generate data using less than 10% of the available cameras; the majority of the rooms have less than two sampled viewpoints out of maximum five. We split our dataset into training, validation, and testing sets of sizes 40k, 5k and 5k, respectively. Figure 4 shows a histogram of

Table 1. **Quantitative evaluation on synthetic scenes.** We report L_2 distances in 2D and 3D and a perceptual loss (top – baselines, bottom – ablations). Notice how our approach out-performs all baselines and ablations across all criteria.

Method	L_2 (2D) $\left(1 \times 10^4\right)$	L_2 (3D) $\left(1 \times 10^4\right)$	Perc. loss $\left(1 \times 10^2\right)$
Dataset prior	616.4	797.8	6.3
Depth + fwdS.	186.5	255.1	4.9
DepthEq + fwdS.	144.1	196.3	4.5
2D regression	4.9	N/A	N/A
3D regression	4.9	6.5	3.1
DepNet	101.2	133.5	2.9
TrajNet	3.3	4.3	2.4
Ours	**1.5**	**2.1**	**1.3**

the span of depth values over the dataset in centimeters, in addition to a set of randomly selected trajectories from the dataset, illustrating the dataset's variety.

5. Experiments

In this section, we show qualitative and quantitative results of our system. To better appreciate our final results, we encourage the reader to view videos of our dynamic composite outputs in the supplemental material.

5.1. Results on synthetic data

As a first experiment, we evaluate the effectiveness of our approach on synthetic scenes from the SUNCG dataset [42], which allows us to directly compare against trajectories resulting from forward simulation.

Dataset and evaluation criteria. We use the generated trajectories resulting from forward simulation as outlined in Section 4. We evaluate our predicted trajectories by comparing against ground truth trajectories using L_2 distance averaged over time. While this criterion evaluates physical accuracy of the predicted trajectory, it does not evaluate the trajectory's visual plausibility. In addition to reporting time-averaged L_2 distance, we also report a perceptual loss over the rendered video by computing the L_2 distance between the image encoding of each frame with the corresponding ground truth frame. We obtain the image encodings through a pre-trained Inception-ResNet model that has been pre-trained on ImageNet [43] and compare the responses from the penultimate layer.

Baselines. We evaluate a number of baselines for our task. First, we consider a baseline (*Dataset prior*) where we compute the average 3D forward trajectory over the training set. Second, we train a neural network to regress to 2D and 3D trajectories (*2D regression* and *3D regression*) given the input single image. For fair comparison, the networks share the same architecture as the trajectory update network outlined in Section 3.1 except the sampled value z is withheld as input; for 2D, we used only two dimensions for input/output. We trained the networks for the same number of epochs while monitoring the validation loss to avoid overfitting. Third, we consider a baseline of a forward physical simulation of the ball using geometry from the predicted depth (*Depth + fwdS.*). We used the same depth-prediction algorithm [8] as in our proposed method. Finally, we consider performing histogram equalization over histogrammed ground truth and predicted depth values over the training set (*DepthEq + fwdS.*).

Ablations. We consider the following ablations of our model. First, we consider running our full pipeline without the depth correction network (*TrajNet*). Second, we consider running our full pipeline without the last trajectory update network (*DepNet*).

Results. To evaluate the improvement we achieve with our depth correction network, in addition to standard L_2 measures, we computed the difference in time to the first bounce event (lower is better): Dataset prior – 3.8, Depth + fwdS. – 3.0, DepthEq + fwdS. – 2.6, DepNet – 1.8; our depth correction network outperforms the baselines on this criterion. We show final quantitative results in Table 1. Notice how we outperform all baselines and ablations across all criteria.

5.2. Results on real data

As a second experiment, we evaluate the effectiveness of our approach on single still images depicting real scenes. As per the synthetic experiments, we trained our system on the synthetic data from Section 4.

Dataset, baselines, ablations. We collected a dataset of 30 *in the wild* natural images depicting indoor scenes from royalty-free sources and compared our approach against the baselines and ablations described in Section 5.1. Additionally, we randomly selected a set of 20 NYUv2 images [41] and compared our approach against forward simulation on the provided depths from an active-sensing camera. Lastly, we compared our method to the work described in [38].

Qualitative results. We show qualitative results in Figure 5. Notice how our approach generally improves over the initial trajectories and out-performs the 3D trajectory regression baseline that returns trajectories close to the mean.

User study. As we do not have noise-free ground truth trajectories for either sets of images, we conducted a user study where we asked humans to judge the visual plausibility of the outputs. More specifically, we presented a user with two outputs from different systems and asked the user to choose which output looks more realistic. We randomized the order in which we showed each output to the user. The experiments were conducted with workers from thehive.ai.

For the set of 30 natural images, we compared our results against forward simulation on predicted depth (Depth

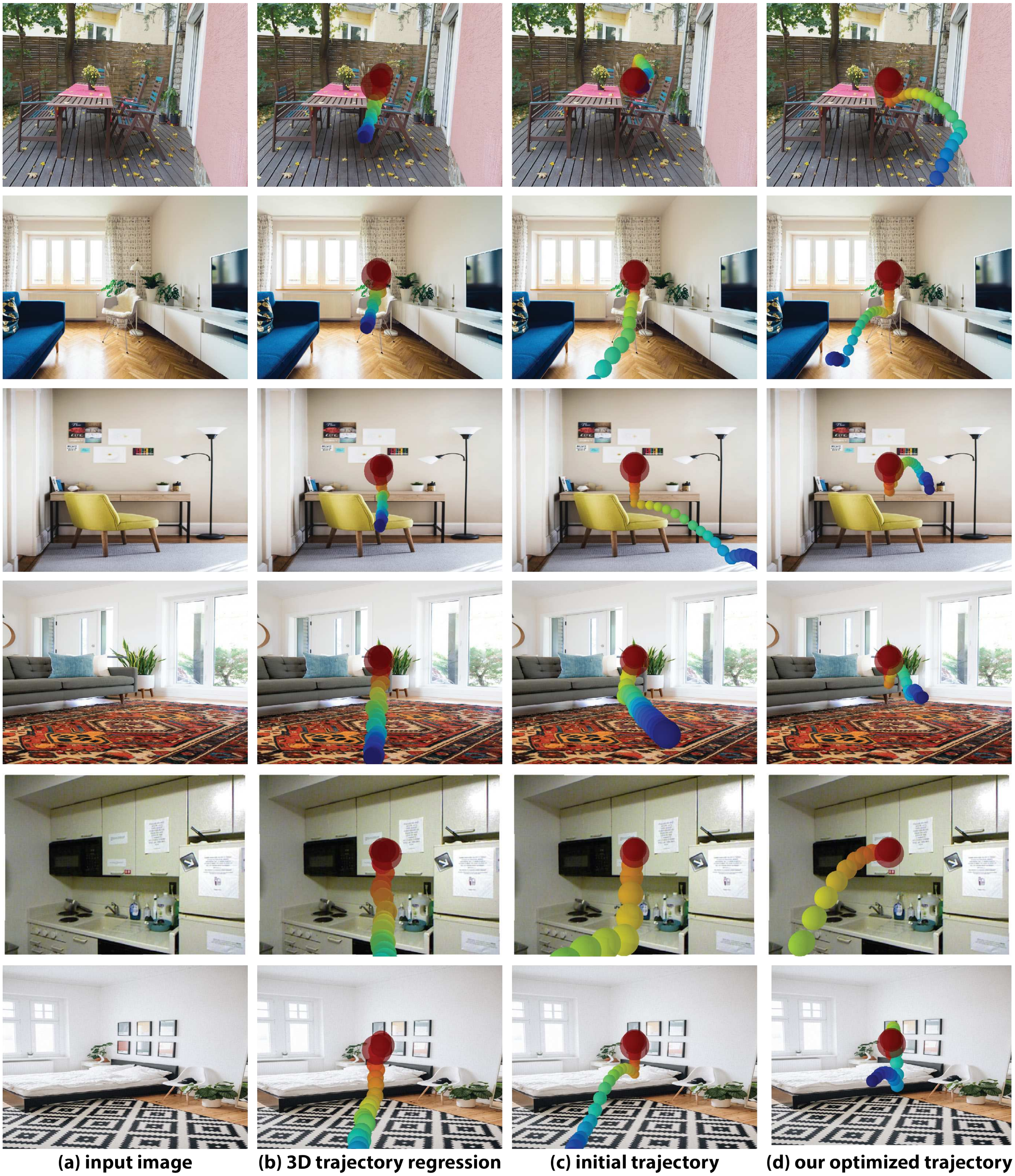

Figure 5. **Qualitative results on real images.** We show (a) the input image depicting a real scene, (b) output from the 3D trajectory regression baseline, (c) our initial trajectory resulting from forward simulation on predicted depth, and (d) our output optimized trajectory. Notice how the visual plausibility of our output trajectories improve over the initial trajectory. Last row – failure example.

+ fwdS.) and our full pipeline without the last trajectory update network (DepNet). Additionally, we also compared Depth + fwdS. against DepNet. Each experiment was conducted by 80 unique users and the users casted 4.5k votes over the three tasks. Users preferred our method over Depth + fwdS. and DepNet 71% and 59% of the time ($p < .0001$ – all p-values from binomial test), respectively, illustrating the effectiveness of our approach. Moreover, users preferred DepNet over Depth + fwdS. in 63% of the cases ($p < .0001$), illustrating that the depth correction network helps improve results. For the NYUv2 images, we evaluated our method against forward simulation on the provided active-sensing depths over 98 unique users casting 2k votes. Users preferred our method 49% of the time (no statistical significance), demonstrating the effectiveness of our approach and the level of noise in the active-sensing depths.

Finally, we compared to the work of [38]. As a direct comparison with [38] is not feasible due to the additional information it requires, we designed a *ground-truth-augmented version* of the method. The work in [38] requires a ground-truth input trajectory up to the first bounce, which we can provide by running PyBullet with appropriate parameters over the geometry obtained from the kinect depths of the NYUv2 dataset. Further, as [38] outputs a post-bounce trajectory spanning only 0.1s and our approach outputs multiple bounces and roll-out trajectories over 1.5s, we manually extended the output from [38] to 1.5s via 3D parabola fitting (note that this step is comparable to simulating post-bounce free fall). We show a qualitative comparison of the resulting trajectories in Figure 6.

For the experiment conducted on the aforementioned data, 130 users cast a total of 2.5k votes. Users preferred our method in 59% of the cases ($p < .0001$), which is noteworthy given that [38] has access to ground-truth about when and where in the scene the bounce occurs and indirect access to kinect depths for the scene. Additionally, we performed a second study, where our trajectories were extended after the first bounce through free fall – note that this is an ablated version of our method as it lacks multi-bounces and roll-outs. For this study, 90 users cast a total of 1.8k votes. Users preferred our outputs 56% of the time ($p < .0001$).

6. Conclusion

We introduced neural re-simulation as a 'correction' mechanism that learns to generate visually plausible bounce interactions of a virtual ball in depicted scenes in single still images. Our system learns to update an initial depth estimate of the depicted scene through our depth correction network and uses this update to correct an initial trajectory obtained via forward simulation through our trajectory update network. We demonstrated our system on not only synthetic scenes from SUNCG, but also on real images. We showed via a human study that our approach on real images yields outputs that are more visually plausible than baselines. Our approach opens up the possibility of generating more complex interactions in single still images, such as inserting objects with different geometry and physical properties and modifying the depicted environment.

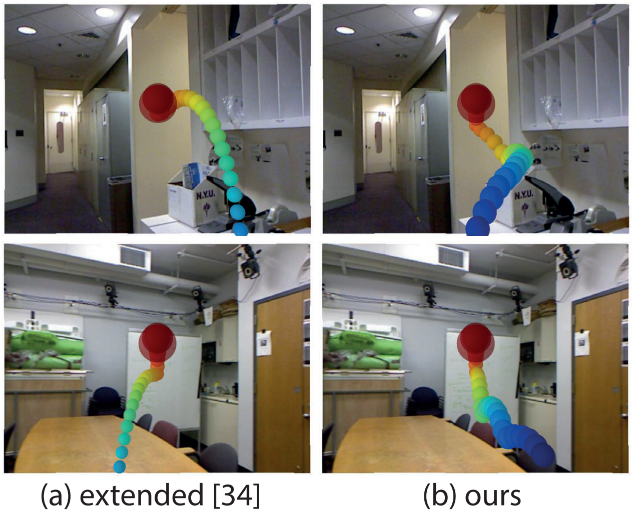

Figure 6. **Sample user study trajectories.** We show our results versus results obtained by providing access to ground-truth depth and extending the work of [38] through free fall. Note that, in ground-truth augmented [38] (see text), the ball passes through scene objects, such as the cubicle (row 1) and table (row 2).

Acknowledgements

We thank our reviewers for their insightful comments. We also thank Senthil Purushwalkam for helping out with the comparison with [38] and Tobias Ritschel, Paul Guerrero and Yu-Shiang Wong for their technical help throughout the project. This work was partially funded by the ERC Starting Grant SmartGeometry (StG-2013-335373).

References

[1] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 3

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 3

[3] Vinay Bettadapura, Caroline Pantofaru, and Irfan Essa. Leveraging contextual cues for generating basketball highlights. In *Proceedings of ACM International Conference on Multimedia (ACM-MM)*. ACM, October 2016. 2

[4] Kiran Bhat, Steven M. Seitz, Jovan Popović, and Pradeep Khosla. Computing the physical parameters of rigid-body motion from video. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2002. 3

[5] Marcus A. Brubaker and David J. Fleet. The kneed walker for human pose tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 3

[6] Marcus A. Brubaker, David J. Fleet, and Aaron Hertzmann. Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision*, 87(140), 2010. 3

[7] Marcus A. Brubaker, Leonid Sigal, and David J. Fleet. Estimating contact dynamics. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2009. 3

[8] Ayan Chakrabarti, Jingyu Shao, and Gregory Shakhnarovich. Depth from a single image by harmonizing overcomplete local network predictions. In *NIPS*, 2016. 3, 6

[9] Michael B. Chang, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 3

[10] Yu-Wei Chao, Jimei Yang, Brian Price, Scott Cohen, and Jia Deng. Forecasting human dynamics from static images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[11] Erwin Coumans and Yunfei Bai. pybullet, a Python module for physics simulation for games, robotics and machine learning. `http://pybullet.org/`, 2016–2017. 2, 3, 5

[12] Abe Davis, Justin G. Chen, and Frédo Durand. Image-space modal bases for plausible manipulation of objects in video. *ACM Trans. Graph.*, 34:239:1–239:7, 2015. 1, 3

[13] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision (IJCV)*, 51(2):91–109, 2003. 2

[14] Sébastien Ehrhardt, Aron Monszpart, Niloy J. Mitra, and Andrea Vedaldi. Learning a physical long-term predictor. *CoRR*, abs/1703.00247, 2017. 3

[15] Sébastien Ehrhardt, Aron Monszpart, Andrea Vedaldi, and Niloy J. Mitra. Learning to represent mechanics via long-term extrapolation and interpolation. *CoRR*, abs/1706.02179, 2017. 3

[16] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning predictive visual models of physics for playing billiards. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 3

[17] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *Proceedings of the International Conference On Intelligent Robots and Systems (IROS)*, 2017. 3

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. 2, 5

[19] Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 3

[20] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Trans. Graph.*, 33(4):127:1–127:12, 2014. 1, 3

[21] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Trans. Graph.*, 37(4):163:1–163:14, July 2018. 2

[22] Theodore Kim and John Delaney. Subspace fluid re-simulation. *ACM Trans. Graph.*, 32(4), 2013. 1

[23] Nikolaos Kyriazis, Iason Oikonomidis, and Antonis Argyros. Binding vision to physics based simulation: The case study of a bouncing ball. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2011. 3

[24] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 430–438. JMLR.org, 2016. 3

[25] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 2016. 3

[26] Richard Mann, Allan Jepson, and Jeffrey Siskind. The computational perception of scene dynamics. In *CVIU*, 1997. 3

[27] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. Lookingood: Enhancing performance capture with real-time neural re-rendering. *ACM Trans. Graph.*, 37(6):255:1–255:14, Dec. 2018. 2

[28] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 3

[29] Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[30] Aron Monszpart, Nils Thuerey, and Niloy J. Mitra. SMASH: Physics-guided reconstruction of collisions from videos. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2016. 3

[31] Roozbeh Mottaghi, Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. Newtonian image understanding: Unfolding the dynamics of objects in static images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[32] Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. "What happens if..." Learning to predict the effect of forces in images. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 3

[33] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. 1

[34] Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. *2011 International Conference on Computer Vision*, pages 2320–2327, 2011. 1

[35] Andrew Owens, Phillip Isola, Josh McDermott, Antonio Torralba, Edward Adelson, and William Freeman. Visually indicated sounds. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[36] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 3

[37] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. In *Proceedings of the International Conference On Robotics and Automation (ICRA)*, 2016. 3

[38] Senthil Purushwalkam, Abhinav Gupta, Danny Kaufman, and Bryan Russell. Bounce and learn: Modeling scene dynamics with real-world bounces. In *International Conference on Learning Representations (ICLR)*, 2019. 3, 5, 6, 8

[39] Syuhei Sato, Yoshinori Dobashi, and Tomoyuki Nishita. Editing fluid animation using flow interpolation. *ACM Trans. Graph.*, 37(5), 2018. 1

[40] Arno Schödl, Richard Szeliski, David Salesin, and Irfan A. Essa. Video textures. In *SIGGRAPH*, 2000. 2

[41] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 3, 6

[42] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 29th IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 5, 6

[43] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 4278–4284, 2017. 4, 6

[44] Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. IGNOR: image-guided neural object rendering. *CoRR*, abs/1811.10720, 2018. 2

[45] Nils Thuerey. Interpolations of smoke and liquid simulations. *ACM Trans. Graph.*, 36(1), 2016. 1

[46] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to Generate Long-term Future via Hierarchical Prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 3

[47] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[48] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from variational autoencoders. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 3

[49] Jui-Hsien Wang, Rajsekhar Setaluri, Dinesh K. Pai, and Doug L. James. Bounce maps: An improved restitution model for real-time rigid-body impact. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)*, 36(4), July 2017. 3

[50] Nicholas Watters, Andrea Tacchetti, Theophane Weber, Razvan Pascanu, Peter Battaglia, and Daniel Zoran. Visual interaction networks. *CoRR*, abs/1706.01433, 2017. 3

[51] Jiajun Wu, Joseph J. Lim, Hongyi Zhang, Joshua B. Tenenbaum, , and William T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 3

[52] Jiajun Wu, Ilker Yildirim, Joseph J. Lim, William T. Freeman, and Joshua B. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 3

[53] Tianfan Xue, Jiajun Wu, Katherine L Bouman, and William T Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 3

[54] Zhoutong Zhang, Jiajun Wu, Qiujia Li, Zhengjia Huang, James Traer, Josh H. McDermott, Joshua B. Tenenbaum, and William T. Freeman. Generative modeling of audible shapes for object perception. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 3

[55] Yixin Zhu, Chenfanfu Jiang, Yibiao Zhao, Demetri Terzopoulos, and Song-Chun Zhu. Inferring forces and learning human utilities from videos. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[56] Yixin Zhu, Yibiao Zhao, and Song-Chun Zhu. Understanding tools: Task-oriented object modeling, learning and recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3