STReSSD: Sim-To-Real from Sound for Stochastic Dynamics

Carolyn Matl[1,2], **Yashraj Narang**[2], **Dieter Fox**[2,3], **Ruzena Bajcsy**[1], **Fabio Ramos**[2,4]
University of California, Berkeley[1], NVIDIA[2], University of Washington[3], University of Sydney[4]
{cmatl,bajcsy}@eecs.berkeley.edu, {ynarang,dieterf,ftozetoramos}@nvidia.com

Abstract: Sound is an information-rich medium that captures dynamic physical events. This work presents STReSSD, a framework that uses sound to bridge the simulation-to-reality gap for stochastic dynamics, demonstrated for the canonical case of a bouncing ball. A physically-motivated noise model is presented to capture stochastic behavior of the balls upon collision with the environment. A likelihood-free Bayesian inference framework is used to infer the parameters of the noise model, as well as a material property called the coefficient of restitution, from audio observations. The same inference framework and the calibrated stochastic simulator are then used to learn a probabilistic model of ball dynamics. The predictive capabilities of the dynamics model are tested in two robotic experiments. First, open-loop predictions anticipate probabilistic success of bouncing a ball into a cup. The second experiment integrates audio perception with a robotic arm to track and deflect a bouncing ball in real-time. We envision that this work is a step towards integrating audio-based inference for dynamic robotic tasks.

Keywords: Sim2Real, Sound, Bayesian Inference, Stochastic Simulation

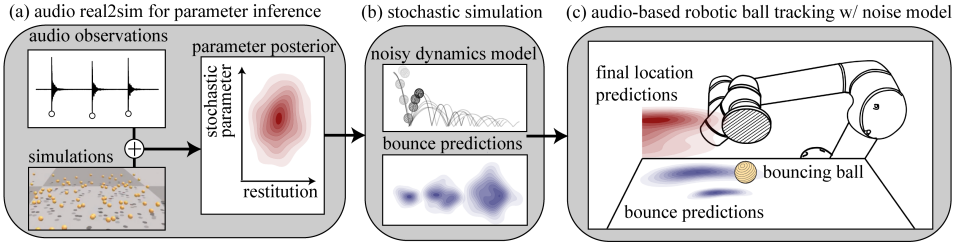


Figure 1: STReSSD is a framework that uses sound to bridge the sim-to-real gap for a stochastic dynamic process (e.g., a bouncing ball). (a) Simulation is used to learn a conditional density function relating a material and noise parameter to collision times and positions, and posteriors over these parameters are generated from audio observations of bouncing balls. (b) A probabilistic dynamics model is learned via the calibrated stochastic simulator. (c) This model is used with real-world auditory feedback for a reactive robotic ball-tracking task.

1 Introduction

The sensory modality of sound can impart practical physical intuition of the surrounding world. In particular, sounds arising from the dynamic interaction of objects or substances through direct contact are highly informative. These sounding interactions enable humans to accomplish various system identification tasks, such as discerning the size [1], length [2], and material [3] of a struck object. The abundance of information encoded in event-driven sound alone calls for the development of robotic techniques to extract physical understanding from this evidently descriptive modality.

Thus, auditory perception presents a natural sensory choice to infer properties of and track an object that is dynamically interacting with its environment. State-of-the-art work has accomplished impressive results in dynamic object-tracking with visual input [4, 5, 6]. However, visual techniques are known to be sensitive to occlusions, specularities, transparencies, and poor lighting conditions. This work serves to complement visual methods with audio perception, which is robust to all these

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

challenges. Furthermore, sound intrinsically is a lower-dimensional and higher-frame-rate signal, which consequently makes it much more computationally suitable for real-time robotic systems.

A canonical dynamical system that is of interest to both the physics and robotics communities is the bouncing ball. Indeed, a wealth of prior work in the physics community has utilized impact sounds to estimate a ball's coefficient of restitution, a parameter that abstracts the energy dissipation upon collision [7, 8, 9]. Nevertheless, it is widely acknowledged that modeling a ball's dynamics with a constant estimate of restitution does not realistically capture its stochastic behavior while interacting with the environment. Notably, Heckel et al. [9] remarks that "even tiny deviations of the shape from the perfect sphere may lead to substantial errors," specifically, due to directional perturbations upon collision (e.g., a tennis ball dropped on asphalt will likely veer off to a side). Prior work has not addressed this stochastic behavior, yet there is evidently a need for modeling these perturbations.

In robotics, simulations are often used to train robotic tasks with the assumption that the simulator models realistic dynamics, yet simulations are inherently deterministic. However, to capture realistic stochasticity within simulation, three challenges come to mind: 1) How does one automatically infer these stochastic parameters? 2) How can they be incorporated into a simulator? and 3) How can this now-stochastic simulator be utilized in a practical way for real robotic tasks?

Contributions: As more research within robotics seeks to bridge the gap between simulation and reality, this work contributes to the community in three specific ways (Figure 1). First, we present a methodology to automatically calibrate simulators from simple features of sound. Second, we introduce learned stochasticity into a robotics simulator, in which a physics-inspired noise model accounts for real-world distributions observed in a bouncing ball's dynamics, e.g., due to uneven or rough surfaces of the ground or ball. Finally, we conduct experiments to demonstrate the predictability of the calibrated simulator in new scenarios, as well as the use of auditory feedback to achieve reactive control in a robotic ball-tracking task. This work combines techniques from multiple fields so that the resulting contributions may enhance the perceptual intelligence of robots using sound.

2 Related Work

2.1 Auditory Perception for Robotic Manipulation

Despite its ubiquity and low cost of capture and processing, auditory perception is not often utilized in real-time robotic manipulation. Instead, it has been most widely used for sound localization [10, 11] and object or material classification [12, 13, 14, 15, 16, 17, 18]. However, only few works have successfully employed it for manipulation tasks. Gandhi et al. [19] showed that sound can be informative of actions applied to sounding objects and thus can be used to predict forward models of the objects. Liang et al. [20] and Clarke et al. [21] also demonstrate dynamic inference from sound by using it for feedback control while pouring liquids and grains, respectively. In contrast, in this work, sound is additionally used to learn physically-relevant parameters, which then, combined with sound as feedback, inform fast robotic interactions with a bouncing ball.

2.2 Simulation Calibration of Dynamics Parameters

A growing field in robotics lies in estimating the simulation-to-reality gap. Domain randomization is a popular technique used to learn policies in simulation that are transferable to the real world [22, 23]. Ramos et al. [24] develops BayesSim, a likelihood-free Bayesian approach to generate mixture-of-Gaussian posteriors over simulation parameters that are used for domain randomization. In general, BayesSim is more sample-efficient and generates better approximations of the posteriors than other likelihood-free inference techniques such as approximate Bayesian computation methods [25]. Recently, BayesSim has been successfully applied in different robotic contexts – for instance, to calibrate granular material simulations, which can be used to predict real-world behavior [26].

This work leverages BayesSim to approximate a posterior over a material and noise parameter, which is sampled for domain randomization when simulating a bouncing ball. Both [27] and [28] use real2sim methods on video input to infer bouncing ball parameters. Our work is unique in that sound is used in lieu of vision, and a noise model is learned to account for observed stochasticity. Learning stochastic parameters has been shown to be useful in works such as [29, 30, 31], where probabilistic models are used to make predictions of planar pushing with frictional contact. The only known works other than this paper that use sound for real2sim focus on realistic sound syn-

thesis [32]. However, Zhang et al. [32] acknowledge their limitations in sim2real transfer, as they rely on bridging the gap using high-dimensional spectral features of sound. In contrast, this paper extracts simple features from audio, making it robust to discrepancies in vibration representations in simulation and real life as well as more suitable for real-time applications.

2.3 Robotic Bouncing Ball Tracking

A bouncing ball is difficult to track due to uncertainties in collision dynamics. However, robot interactions with a dynamic object require accurate tracking and trajectory predictions. Several works have attempted bouncing-ball tracking with vision, either by using high-speed or multiple cameras [5] or pairing low-cost cameras with sim2real tuning of physically-relevant parameters [27, 28]. By using dynamics models trained in the tuned simulator, trajectory predictions compensate for tracking uncertainties [28]. In contrast, this work presents ball tracking through sound and physical parameter inference. The only known work that uses sound to interact with a bouncing ball is [33], where the author designed a mechanism that continuously bounced a ball off an actuated platform. However, sound is used reactively, rather than to inform predictions about the ball dynamics.

3 Methodology

The STReSSD framework combines BayesSim [24] with audio processing to calibrate a stochastic dynamic simulator. We highly recommend referring to Appendix A, which illustrates an expanded version of the STReSSD framework. The following sections detail each module in the framework.

3.1 Simulation Calibration with BayesSim

This work aims to calibrate a physics simulator using auditory signals of ball-table collisions to closely predict real-world dynamics of the ball. Let θ be the simulation parameters to be inferred, and let X represent information-rich features of an auditory signal. Assuming that there is an equivalent representation of X in simulation (X^s) and the real physical environment (X^r), then the problem can be formulated as computing the posterior $p(\theta|X^r)$ over the simulation parameters. This work uses a likelihood-free framework called BayesSim [24] to approximate this posterior function. Specifically, N pairs of parameters and feature vectors $\{\theta_i, X_i^s\}_{i=1}^N$ are first generated from forward simulations of samples from a physically-motivated prior distribution, $p(\theta)$. A conditional density function $q_\Phi(\theta|X^s)$ is then learned by mapping extracted simulation features X^s to a mixture of K Gaussian components with a mixture density neural network (MDNN), parameterized by Φ. The conditional density function q_Φ, along with a real observation X^r, are used to generate an approximation of the posterior $p(\theta|X^r)$. The main advantage of using BayesSim over non-Bayesian techniques (e.g., classical optimization) is that the approximated posterior can be informative of the uncertainties regarding the parameters θ, as well as sensitivity to measurement noise.

3.2 Stochastic Simulation of a Bouncing Ball

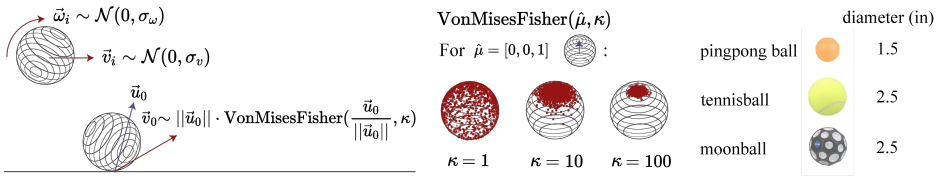


Figure 2: (Left): The initial velocity is sampled from a Gaussian distribution and collision perturbations are sampled from a von Mises-Fisher distribution, centered at $\vec{u}_0/||\vec{u}_0||$. (Middle): 1000 samples over a sphere for different values of κ for $\mu = [0, 0, 1]$. (Right): Balls used in real experiments.

Here, we define the simulation parameters θ that are used to model the stochastic dynamics of a bouncing ball. The coefficient of restitution e is a standard ratio used to quantify energy loss upon object collision. For an object bouncing on an immovable surface, e can be calculated as the ratio of the exiting to entering velocity of the collision. However, as velocities are difficult to extract from sound, bounce times are used to calculate $e = \frac{\text{time from bounce } i \text{ to } i+1}{\text{time from bounce } i-1 \text{ to } i}$ (see Appendix B.1).

While e is generally used as a *constant* in simulation, numerous factors contribute to nontrivial variance in measurements of e [34]. For example, if a ball is dropped vertically onto a level flat surface,

surface asperities cause the ball to experience horizontal perturbations. Because we are measuring e from bounce times instead of velocity, these horizontal perturbations produce a *distribution* over e.

Montaine et al. [34] address this variance by fitting a probability density over e. Instead, this paper aims to represent e and surface asperities as independent variables. To simulate collision perturbations, Montaine et al. [34] represent a ball as a composite multisphere particle with $\sim 10^6$ surface asperities. In contrast, this paper reduces computational complexity by imitating the stochastic dynamics of a bouncing ball with a simple noise model. At every collision, we apply a random perturbation, sampled from a Gaussian distribution on a sphere (i.e., the von Mises-Fisher distribution), to the direction of the outgoing velocity vector. The von Mises–Fisher distribution on a 2-sphere (Figure 2) is given by the probability density $f(x;\mu,\kappa) = \frac{\kappa}{2\pi(e^{\kappa}-e^{-\kappa})}e^{(\kappa\mu^T x)}$ for 3-dimensional random unit vector x, where μ is the mean direction and κ is a concentration parameter.

Let $\vec{u}_0$ represent a ball's unperturbed exit velocity during a collision. To simulate a collision perturbation (e.g., due to rough surfaces), a vector is sampled from $f(x;\hat{\mu},\kappa)$, where $\hat{\mu} = \frac{\vec{u}_0}{||\vec{u}_0||}$. The vector is then scaled by $||\vec{u}_0||$ to produce a perturbed exit velocity $\vec{v}_0$. Thus, the outgoing kinetic energy of the ball is unaffected by the perturbation. Stochastic ball dynamics are therefore modeled by e, for energy dissipation, and κ, to account for the stochasticity due to collisions.

While this model can be integrated with most standard simulators, we used NVIDIA's Isaac Simulator paired with a preconditioned conjugate residual (PCR) solver [35]. This GPU-optimized simulation platform enabled the parallelization of hundreds of environments. This was particularly useful for efficiently generating distributions of ball trajectories due to the modeled collision perturbations.

3.3 Audio Processing

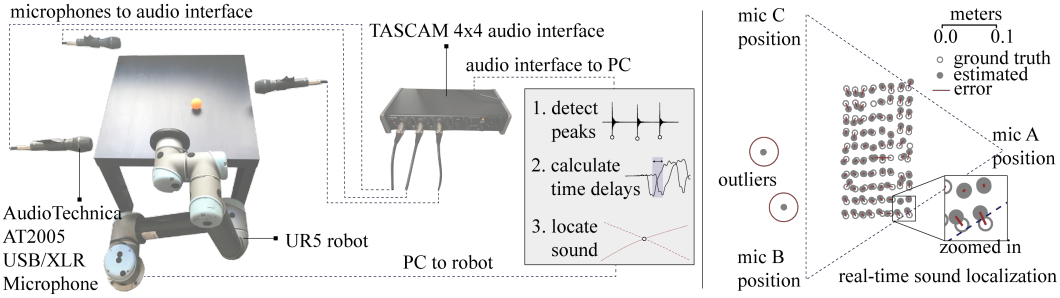


Figure 3: (Left): Integrated robotic system used in experiments. (Right): Real-time sound localization is slightly less accurate than the offline method and produces outliers with large errors.

We hypothesized that the times and positions of the ball bounces are informative of $\theta = \{e, \kappa\}$, as e can be expressed as a function of impact times (Section 3.2), and κ affects the ball bounce positions. This motivates our four-dimensional feature vector X, which encodes both temporal and positional information. Letting t_i, $\vec{p}_i$, and d_i denote the time, positional vector, and distance between bounce i and $i+1$, then X is composed of $\frac{t_2}{t_1}$, $d_1 = ||\vec{p}_1||_2$, $d_2 = ||\vec{p}_2||_2$, and α, the signed angle between $\vec{p}_1$ and $\vec{p}_2$ (see Appendix B). Higher dimensional spectral features were not needed to estimate θ, but future work could leverage this additional information to, for instance, classify the ball's material.

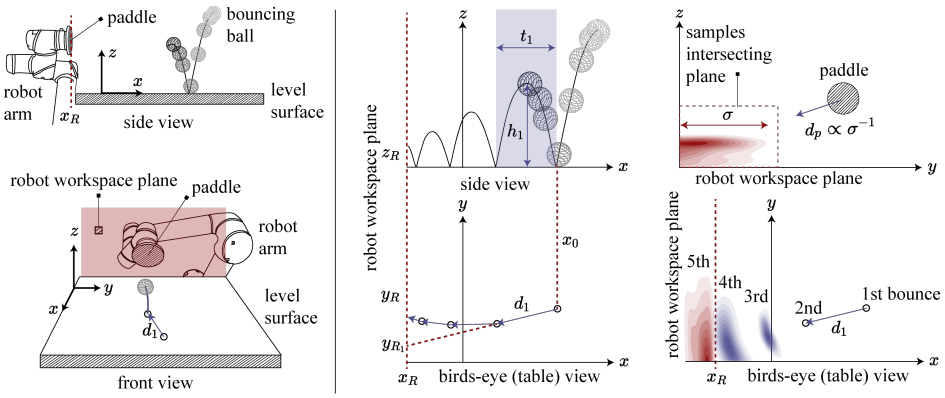
Vector X is simple to extract from simulation, but, in the real-world, depends on accurate sound localization for each bounce. We used two localization methods – an offline method for parameter inference and an online method for real-time interaction, both of which relied on the same principle of time delay estimation (TDE). TDE uses time delays between microphone signals to locate sound sources (see Appendix C for derivation). For 2D localization, only 3 microphones are necessary.

The offline and real-time methods diverged in how these time delays were calculated. Offline time delays were calculated using phase correlation, a standard frequency-domain technique [36], over audio segments of >20 ms. In contrast, real-time processing required fast calculations on 11 ms-length buffers of incoming audio. Peak times corresponding to the strong peaks induced by collisions could be detected efficiently and thus were used to calculate time delays. The offline method was found to be slightly more accurate, with an average error of about 6.7 mm compared to real-time's 7.9 mm, shown in Figure 3 (see Appendix C for comparisons). Experiments were conducted in a household environment rather than a sound studio, so real-time localization was slightly less robust.

The method's sensitivity to reverberation, or echoes from early reflections off of walls and furniture, occasionally caused large errors in detecting true impact times. However, the performance was a necessary trade-off to enable reactive robot interactions. Furthermore, peak detection was generally unaffected by other factors such as noisy robot motion or acoustic differences in ball material.

For experiments, auditory signals were captured by three AudioTechnica AT2005 USB/XLR Microphones, which were placed around a 0.55-by-0.55 meter particle-board table, 3 inches above the table surface. Microphone signals were streamed at 44.1 kHz in parallel to a PC through a TASCAM 4x4 audio interface (Figure 3). For robot experiments, audio was processed in real-time, and control signals were sent to a UR5 robot in parallel processes. The overall latency of the system was dominated by robot motion, as bounce localization and inference ran at $\sim$100 and 20 Hz, respectively.

3.4 Robotic Ball Tracking



(a) experimental setup for ball-tracking task (b) dynamic predictions without noise (c) dynamic predictions with noise

Figure 4: (a) Ball tracking setup. (b) Example ball trajectory for noiseless method. y_{R_1} is the y-coordinate of the inferred crossing point of the ball given the first two bounces. (c) Samples of future bounce positions given the first two bounces using the stochastic noise method. The mean and variance of the samples in the workspace plane inform the distance the paddle should travel (d_p).

To illustrate the benefits of modeling dynamic stochasticity and the real-time capabilities of audio perception, we compare two methods to accomplish the robotic task of tracking and deflecting a bouncing ball using sound. The methods differ in their consideration of perturbations upon collision. Pseudocode and equations accompanying each method are presented in Appendices D and E.

3.4.1 Deterministic Baseline: Dynamic Predictions without Collision Noise

Without modeling collision perturbations (thus ignoring concentration parameter κ), the dynamics of a bouncing ball are defined only by its initial conditions, gravity, and energy-dissipating forces. For this particular method, we assume that the most significant parameter affecting ball dynamics is e. The ball trajectory can then be defined by classical projectile equations, with e applied to simulate energy dissipation at each bounce. Figure 4a-b illustrate the control scheme used to guide the robot to contact the ball as it enters the workspace plane. The ball's future bounces can be extrapolated from the times and positions of the last two bounces, since here we assume that the ball is not directionally perturbed. The goal position of the robot, denoted as (x_R, y_R, z_R) in Figure 4, is then defined as the intersection of the piecewise parabolic trajectories with the robot plane. If the ball will intersect the plane at t_R seconds before the next bounce, then the robot moves to the goal position in that time. Otherwise, the robot moves to y_R before the next bounce is observed.

3.4.2 Stochastic Method: Dynamic Predictions with Collision Noise

Predicting the dynamics of a ball without incorporating knowledge of collision noise may cause the robot to be over-reactive to small perturbations in the ball's trajectory. Furthermore, success relies on accurate real-time sound localization, which as mentioned earlier, can occasionally be sensitive to reverberations, causing large errors in the inferred bounce location (Figure 3). Thus, we aim to incorporate the noise model to address two goals: leveraging the uncertainty of the ball's location to prevent the robot from over-reacting, and filtering unrealistic inferred bounce locations.

Both goals employ a second BayesSim model that generates posteriors over the next bounce time and location, given the time and distance between the prior two bounces. The BayesSim model, trained via the calibrated simulator, serves as a probabilistic dynamics model. At run-time, a generated posterior is used with an elliptic envelope filter to classify if the observed bounce can be trusted (or if it is an outlier). If the latter, the mean of the posterior is used to estimate the bounce location.

In addition, the ball's future trajectory is estimated by generating samples from the posterior, which in turn are used to generate new posteriors. Thus, trajectory predictions are made by recursively generating posteriors until samples intersect the plane. In this new context, BayesSim is used as a generic regression technique to learn a transition model, which is applied recursively, and whose predictions are averaged. This dynamics model could also be incorporated into a nonlinear filtering technique such as an Unscented Kalman Filter, although this would require defining a sensor model. Figure 4c shows that the predicted future bounce location uncertainty grows with every bounce. This uncertainty informs the end-effector motion. The smaller the variance of the predictions are, the closer the end-effector will move toward the goal location before the next observation.

4 Experiments

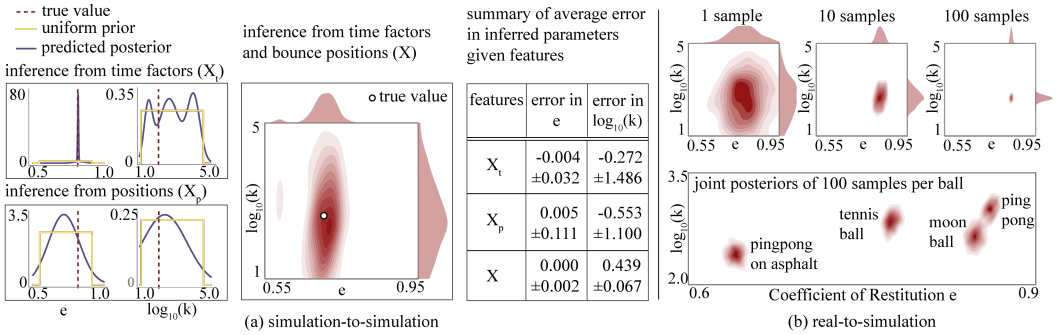


Figure 5: (a) Sim-to-sim example posteriors of inferring e and $\log_{10}(\kappa)$ from bounce times, e and $\log_{10}(\kappa)$ from bounce positions, and both e and $\log_{10}(\kappa)$ from times and positions. The table summarizes average error over 1000 examples for each parameter, given a feature subset. (b) Top: Real-to-sim posteriors become more certain with more samples. Bottom: Posteriors for real balls.

4.1 Simulation-to-Simulation Experiments

The inference framework was first tested with simulated data to assess whether bounce locations and times were relevant measurements for the inference of the desired model variables (e and κ). Training data was generated in simulation by sampling from a physically-motivated uniform prior distribution, with $e \in [0.55, 0.95]$ and $\log_{10}(\kappa) \in [1, 5]$. Each simulation sampled an initial angular and linear velocity from a Gaussian centered at 0 (see Figure 2) to mimic real-world noise when dropping a ball. 6000 and 1000 simulations were run for the training and testing dataset, respectively.

To measure how informative each feature in X was for each parameter in θ, an ablation study was performed over the simulated dataset. The features were divided into two sets, one relevant to time ($X_t = \frac{t_2}{t_1}$), and the other, to position ($X_p = \{d_1, d_2, \alpha\}$). Four BayesSim models were trained, each using either X_t or X_p to infer one of e or κ. These models were used to generate posteriors from a testing dataset of 1000 examples, and the peaks of the posteriors were compared with the true value of the inferred parameter. The table in Figure 5a summarizes the average error in inferred parameters e and κ, given a subset of features. Posteriors that are unimodal and low-variance have higher certainty and thus relevance to the inferred parameter. As reflected in the single-parameter posteriors of Figure 5a, time appears to be a more informative observation for e, while positional information is more relevant for κ. Knowledge of both time and positional information generates significantly more certain posteriors. Observe that the variance of the marginal posterior for $\log_{10}(\kappa)$ will naturally be greater than that of e, as estimating a noise parameter is inherently more difficult.

4.2 Real-to-Simulation Experiments

With the assurance that X contained informative features for θ, we turned to extracting θ to describe real bouncing balls. Note that both e and κ are reflective of material-to-material interactions, and

thus are unique to each pair of object and surface. The four object-surface pairs tested were a ping-pong ball, tennis ball, and moon ball (which has large crater-like divots) bouncing on a smooth table, and a ping-pong ball bouncing on an asphalt surface (see Figure 2). This was to demonstrate the effect of surface asperities both on the ground (asphalt) as well as on the ball (e.g., moon ball).

For each ball-surface pair, the ball was dropped 100 times from 0.26 meters (to match simulation), and offline sound localization captured the locations and times of the bounces. Real-world feature vector X^r was extracted for each drop and used to generate a posterior over θ. Figure 5b shows an example posterior generated from one sample, or drop, of the moon ball. Note that, from one sample, the posterior is quite uncertain. However, assuming independence between each new observation X_i^r, the joint posterior can be generated by multiplying each individual posterior together. As the single sample posteriors reflected strongly Gaussian structures, each mixture-of-Gaussians was projected to a single Gaussian before multiplication. As shown in Figure 5b, the joint posterior became more certain with more observations, with the joint posterior of 100 samples demonstrating strongly peaked marginals in both e and $\log_{10}(\kappa)$. The resulting posteriors of each ball-surface pair are illustrated in Figure 5b, and their positions relative to each other qualitatively reflect their real-world dynamic characteristics (e.g., the moon ball is bouncier and noisier than the tennis ball).

4.3 Closing the Loop: Real-to-Sim-to-Real

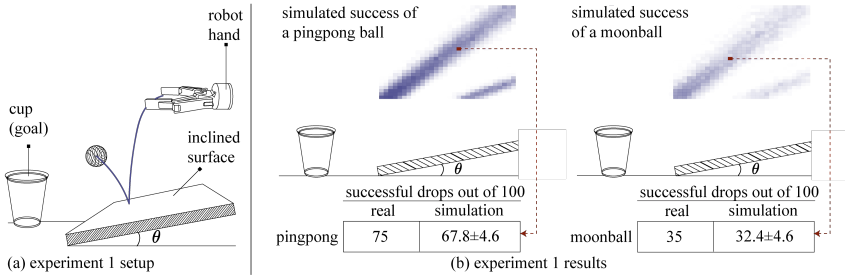


Figure 6: (a) A robotic hand bounces the ball off the tilted table surface into the cup, or goal. (b) Top: Spatial map of success rates for different initial positions of the ball drop. Success ranges from lightest (0 successful) to darkest (100 successful). Bottom: Comparison of real and simulated success rates (out of 100) dropped at the red point, with simulated success rates averaged over 30 trials of 100 simulations.

In order to test if the dynamic model of the bouncing ball generalized well to new scenarios, an experimental setup similar to the one constructed in Allevato et al. [27] was used, as illustrated in Figure 6. The table surface used in Section 4.2 was tilted at an angle of $10.72\,\mathrm{deg}$, and a plastic cup was placed 8.6 inches away from the bottom of the incline. The physical experimental setup was replicated in simulation, and the goal was to compare the success rate of bouncing a ball against the inclined plane into the cup in simulation and in real experiments. While a deterministic simulator would only either fail or succeed if a ball was dropped at a specific initial position, by modeling dynamic noise due to surface perturbations, our stochastic simulator had probabilistic success.

Using a Robotiq 2F-140 gripper to drop the ball repeatedly over the inclined surface at a fixed position, real experiments showed similar probabilistic levels of success, with the pingpong ball and moon ball bouncing into the cup (out of 100) 75 and 35 times, respectively. The experiment was repeated in simulation 30 times, for each ball and their respective drop positions. For each simulation, a new pair of e and $\log_{10}(\kappa)$ were sampled from the inferred posteriors from Section 4.2. The average simulated success rates were within close range of the real success rates, suggesting that the learned dynamics model could generalize well to novel physical scenarios. This also implies that the simulator could potentially be used for predictive measures, e.g., searching for where to drop a ball to maximize success rate. To demonstrate this concept, we sampled success rate (out of 100) over a discrete number of initial positions in simulation. This generated a spatial map of probabilistic success (Figure 6), a capability made possible through the modeling of collision perturbations.

4.4 Robotic Ball-Tracking Results

The goal of the ball-tracking task was to move the robot end-effector, a circular paddle with a diameter of 4.5 inches, to touch the bouncing ball as it entered the robot workspace plane, using only sound as feedback. Each of the methods described in Section 3.4 was evaluated for the tennis

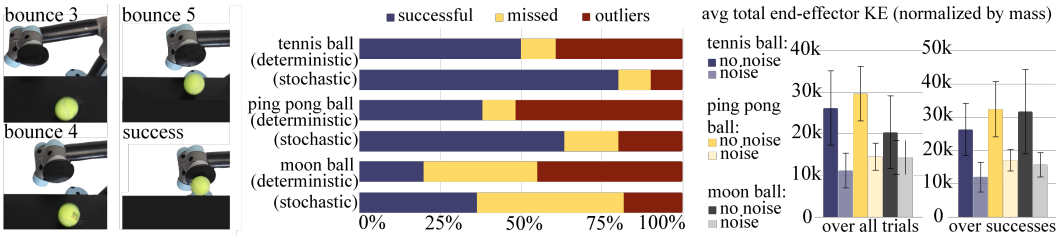


Figure 7: (Left): A successful trial. (Middle) "Outliers" signifies failures due to sound localization errors, and "Missed" denotes any other failure. Compared to the deterministic baseline, the stochastic method has higher success (an absolute increase of 24%) and fewer failures due to outliers (an absolute decrease of 29%), averaged across all balls. This graph excludes trials that violated the robot's boundary constraints; an alternative graph with all 30 trials can be found in Appendix E.1. (Right): On average, the baseline uses more total energy than the stochastic method over all trials and all successful trials, suggesting that the baseline is more overreactive.

ball, ping-pong ball, and moon ball on the following metrics: 1) success out of 30 trials (whether or not the end-effector touched the ball), and 2) kinetic energy of the end-effector, integrated over the trajectory. We hypothesized that, by modeling collision perturbations, the probabilistic method would be more robust to errors in bounce localization as well as curb the reactiveness of the robot to small perturbations in the ball's trajectory. Both methods assumed nothing about the initial condition of the ball, and for each trial, the ball was manually tossed onto the table in the direction of the robot.

As shown in Figure 7, the stochastic method averaged an absolute increase in success rate of 24%, which corresponds to an improvement of 70%, relative to the deterministic baseline. Failure cases are labeled as "outliers" for errors in bounce localization and "missed" for all other failures. For each of the balls, a few of the 30 tosses resulted in the ball violating the robot workspace boundaries, so those tosses were removed from analysis. Sensitivity to "outliers" in bounce localization can be resolved by using dynamic predictions with collision noise described in Section 3.4.2. Figure 7 precisely demonstrates this effect, where the number of failure cases due to outliers in bounce localization diminished for all three balls. While the stochastic method eliminates failures due to outliers in later bounces, it is still sensitive to outliers in the first and second bounces, since it relies on those to generate posteriors for future bounces. Finally, Figure 7 shows that the average total end-effector kinetic energy over all trials, normalized by mass and averaged across balls, was reduced by 46% when considering collision noise versus without. This pattern is replicated when looking at the average energy over just the successful trials (a reduction by 51%), suggesting that the stochastic method does indeed curb over-reactions to small perturbations in a bouncing ball's trajectory.

5 Conclusion

This work presents STReSSD, a novel framework for calibrating a simulator for stochastic dynamics from simple, low-dimensional features of sound. A physically-motivated noise model is learned from real audio, which enables a physics simulator to reflect real stochastic dynamics caused by rough or uneven surfaces. Using the canonical dynamical system of a bouncing ball, this work closes the loop on real-to-sim-to-real by demonstrating the predictive capabilities of the learned dynamics model in both an open-loop and sound-in-the-loop robotics task. To the best of our knowledge, this is the first robotics work that relies solely on sound to accomplish a highly dynamic, probabilistic task. We believe that the robotic capabilities demonstrated by the devised framework highlight the utility of audio perception and may encourage increased integration of sound in robotics.

While the experimental results are promising, there are various opportunities to extend this work. Sensor fusion with vision and more robust peak-detection can improve the accuracy of real-time sound localization. Furthermore, one could imagine utilizing the rich spectral information from the raw audio signals to classify the material of the ball. The multivariate classification uncertainty could then be incorporated into the BayesSim model to produce a multimodal posterior distribution over simulation parameters. Future work aims to explore simulation-in-the-loop inference, where the simulator is calibrated in real-time and a dynamic object is tracked using a particle filter or an Unscented Kalman Filter. These advancements may lead to broader impact in various sub-fields of robotics, including navigation and unseen-object manipulation.

Acknowledgments

The authors were supported in part by the National Science Foundation Graduate Research Fellowship. We thank Matthew Matl and all reviewers for their insightful feedback and suggestions.

References

[1] M. Grassi. Do we hear size or sound? Balls dropped on plates. *Perception & psychophysics*, 67(2):274–284, 2005.

[2] C. Carello, K. L. Anderson, and A. J. Kunkler-Peck. Perception of object length by sound. *Psychological science*, 9(3):211–214, 1998.

[3] R. L. Klatzky, D. K. Pai, and E. P. Krotkov. Perception of material from contact sounds. *Presence: Teleoperators & Virtual Environments*, 9(4):399–410, 2000.

[4] P. Cigliano, V. Lippiello, F. Ruggiero, and B. Siciliano. Robotic ball catching with an eye-in-hand single-camera system. *IEEE Transactions on Control Systems Technology*, 23(5):1657–1671, 2015.

[5] S. Gomez-Gonzalez, Y. Nemmour, B. Schölkopf, and J. Peters. Reliable real-time ball tracking for robot table tennis. *Robotics*, 8(4):90, 2019.

[6] H. Li, H. Wu, L. Lou, K. Kühnlenz, and O. Ravn. Ping-pong robotics with high-speed vision system. In *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 106–111. IEEE, 2012.

[7] A. D. Bernstein. Listening to the coefficient of restitution. *American Journal of Physics*, 45 (1):41–44, 1977.

[8] I. Stensgaard and E. Lægsgaard. Listening to the coefficient of restitution-revisited. *American Journal of Physics*, 69(3):301–305, 2001.

[9] M. Heckel, A. Glielmo, N. Gunkelmann, and T. Pöschel. Can we obtain the coefficient of restitution from the sound of a bouncing ball? *Physical Review E*, 93(3):032901, 2016.

[10] J. C. Murray, H. R. Erwin, and S. Wermter. Robotic sound-source localisation architecture using cross-correlation and recurrent neural networks. *Neural Networks*, 22(2):173–189, 2009.

[11] S. Lang, M. Kleinehagenbrock, S. Hohenner, J. Fritsch, G. A. Fink, and G. Sagerer. Providing the basis for human-robot-interaction: A multi-modal attention system for a mobile robot. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 28–35, 2003.

[12] C. A. Brooks and K. Iagnemma. Vibration-based terrain classification for planetary exploration rovers. *IEEE Transactions on Robotics*, 21(6):1185–1191, 2005.

[13] J. Christie and N. Kottege. Acoustics based terrain classification for legged robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3596–3603. IEEE, 2016.

[14] E. Krotkov, R. Klatzky, and N. Zumel. Robotic perception of material: Experiments with shape-invariant acoustic measures of material type. In *Experimental Robotics IV*, pages 204–211. Springer, 1997.

[15] O. Kroemer, C. H. Lampert, and J. Peters. Learning dynamic tactile sensing with robust vision-based training. *IEEE transactions on robotics*, 27(3):545–557, 2011.

[16] A. Sterling, J. Wilson, S. Lowe, and M. C. Lin. Isnn: Impact sound neural network for audio-visual object classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 555–572, 2018.

[17] J. Sinapov, C. Schenck, and A. Stoytchev. Learning relational object categories using behavioral exploration and multimodal perception. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5691–5698. IEEE, 2014.

[18] C. L. Chen, J. O. Snyder, and P. J. Ramadge. Learning to identify container contents through tactile vibration signatures. In *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 43–48. IEEE, 2016.

[19] D. Gandhi, A. Gupta, and L. Pinto. Swoosh! rattle! thump!-actions that sound. In *RSS*, 2020.

[20] H. Liang, S. Li, X. Ma, N. Hendrich, T. Gerkmann, F. Sun, and J. Zhang. Making sense of audio vibration for liquid height estimation in robotic pouring. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5333–5339. IEEE, 2019.

[21] S. Clarke, T. Rhodes, C. G. Atkeson, and O. Kroemer. Learning audio feedback for estimating amount and flow of granular material. In *CoRL*, pages 529–550, 2018.

[22] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

[23] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[24] F. Ramos, R. C. Possas, and D. Fox. BayesSim: Adaptive domain randomization via probabilistic inference for robotics simulators. In *RSS*, 2019.

[25] G. Papamakarios and I. Murray. Fast ε-free inference of simulation models with bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, pages 1028–1036, 2016.

[26] C. Matl, Y. Narang, R. Bajcsy, F. Ramos, and D. Fox. Inferring the material properties of granular media for robotic tasks. In *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020.

[27] A. Allevato, E. S. Short, M. Pryor, and A. L. Thomaz. Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. In *Conference on Robot Learning*, pages 445–455, 2019.

[28] M. Asenov, M. Burke, D. Angelov, T. B. Davchev, K. Subr, and S. Ramamoorthy. Vid2Param: Modelling of dynamics parameters from video. *IEEE Robotics and Automation Letters*, 2019.

[29] M. Bauza and A. Rodriguez. A probabilistic data-driven model for planar pushing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3008–3015. IEEE, 2017.

[30] M. Bauza, F. R. Hogan, and A. Rodriguez. A data-efficient approach to precise and controlled pushing. In *Conference on Robot Learning*, 2018.

[31] D. Ma and A. Rodriguez. Friction variability in planar pushing data: Anisotropic friction and data-collection bias. *IEEE Robotics and Automation Letters*, 3(4):3232–3239, 2018.

[32] Z. Zhang, Q. Li, Z. Huang, J. Wu, J. Tenenbaum, and B. Freeman. Shape and material from sound. In *Advances in Neural Information Processing Systems*, pages 1278–1288, 2017.

[33] T. Kuhn. electrondust, Jul 2018. URL https://electrondust.com/2018/07/22/stepper-juggler/.

[34] M. Montaine, M. Heckel, C. Kruelle, T. Schwager, and T. Pöschel. Coefficient of restitution as a fluctuating quantity. *Physical Review E*, 84(4):041306, 2011.

[35] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and V. Makoviychuk. Non-smooth newton methods for deformable multi-body dynamics. *ACM Transactions on Graphics (TOG)*, 38(5):1–20, 2019.

[36] C. D. Kuglin. The phase correlation image alignment methed. In *Proc. Int. Conference Cybernetics Society*, pages 163–165, 1975.

A STReSSD framework

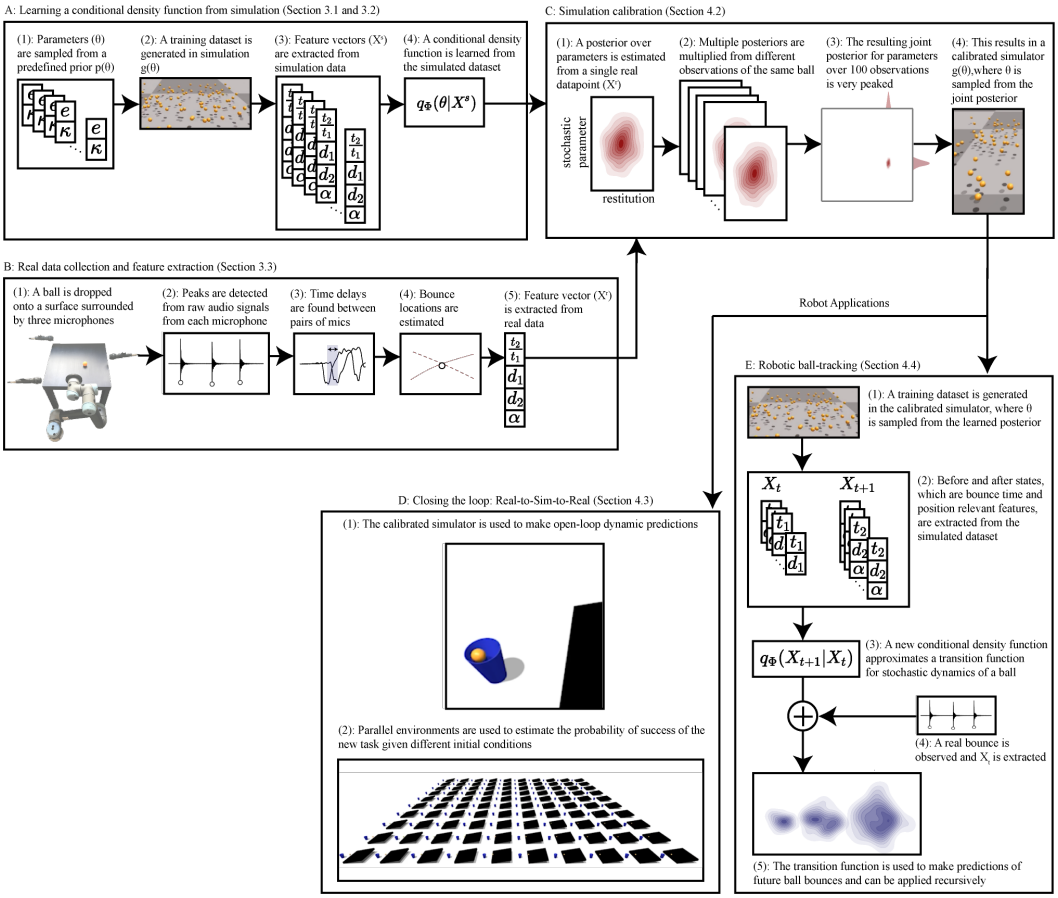


Figure 8: A flow diagram detailing the STReSSD framework. (A): Simulation is used to learn the conditional density function $q_\Phi(\theta|X^s)$ relating parameters e and κ to observation features $X = [\frac{t_2}{t_1}, d_1, d_2, \alpha]$. 6000 independent simulations are run to generate the training set to learn q_Φ. (B): Feature extraction of real data begins by observing the peaks of raw audio signals, finding the time delays between pairs of microphones, and locating the bounces. The low-dimensional feature vector X^r is thus extracted from raw data. (C): The simulator is calibrated using real observations to match a particular ball's dynamic behavior. A posterior over the parameters $\theta = [e, \kappa]$ is approximated using the learned conditional density function q_Φ and the real observation X^r. Posteriors from different observations of the same ball can be multiplied to generate a more peaked joint posterior. The calibrated simulator samples from this joint posterior. (D): The calibrated simulator from (C) is used to make open-loop dynamic predictions of the ball. (E): The calibrated simulator is used for robotic ball-tracking. A new training set is generated by forward simulating the calibrated simulator, with relevant parameters sampled from the joint posterior in (C). New features ($X_t = [t_t, d_t]$ and $X_{t+1} = [t_{t+1}, d_{t+1}, \alpha_{t+1}]$) are extracted from this simulator and a second conditional density function approximates a transition function for the stochastic dynamics of the ball. The transition function is used to estimate the next bounce location and time, given the last two bounces. The transition function can be applied recursively to predict future ball bounces.

B Feature vector X

Feature vector X is composed of four elements. Let X_t refer to the feature in X relevant to time and X_p refer to the three features relevant to position. Below, we derive the relationship between the coefficient of restitution e and feature X_t, as well as how the features in X_p are extracted from the positions of the first three ball bounces.

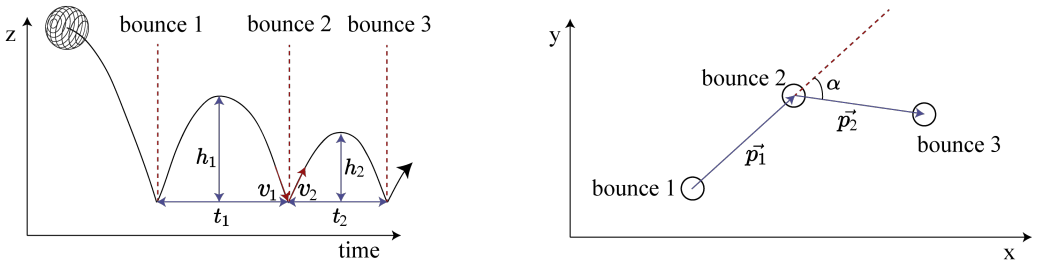


Figure 9: (Left): A graph of the vertical trajectory of a ball over time, labeled with relevant variables for extracting X_t. (Right): xy planar positions of ball bounces are denoted by the open circles, and relevant variables for extracting X_p are labeled.

B.1 Derivation of X_t from e

The coefficient of restitution e is an expression relating the relative kinetic energy of a system before and after a collision. For a bouncing ball on a perfectly rigid surface, e can be expressed by the following ratio:

$$e = \sqrt{\frac{\text{KE}_{\text{after collision}}}{\text{KE}_{\text{before collision}}}} \tag{1}$$

We would like to extract e from audio signals of a bouncing ball, assuming that it is dropped with 0 initial linear velocity. With this assumption, the problem can be simplified to 1-dimension, where the ball travels only in the vertical direction (this is, of course, assuming for now that the ball is not perturbed upon collision). Letting v_1 and v_2 be the ball velocity before and after the 2nd bounce (see Figure 9), then e can be rewritten in terms of the entering and exiting velocities:

$$e = \sqrt{\frac{1/2 m v_2^2}{1/2 m v_1^2}} = \frac{v_2}{v_1} \tag{2}$$

However, velocities are difficult to measure or perceive with sound. Thus, we want to relate e to the times between bounces 1 and 2 (t_1) and bounces 2 and 3 (t_2). Let h_1 and h_2 be defined by the heights depicted in Figure 9. Assuming that, between collisions, the bouncing ball trajectory can be defined by classical projectile equations (assuming an absence of factors such as air resistance), then v_1 can be related to t_1 by the following derivation:

We know from classical projectile equations that:

$$\frac{1}{2}mv_1^2 = mgh_1 \text{ and } \frac{1}{2}g(\frac{t_1}{2})^2 = h_1 \tag{3}$$

Then combining the above equations, we get a relationship for v_1 and t_1:

$$\frac{1}{2}v_1^2 = \frac{1}{2}g^2(\frac{t_1}{2})^2 \rightarrow v_1 = \frac{gt_1}{2} \tag{4}$$

Thus, e can be rewritten in terms of the times between bounces:

$$e = \frac{v_2}{v_1} = \frac{t_2}{t_1} \tag{5}$$

Because we wanted to infer the parameter e, we chose $X_t = \frac{t_2}{t_1}$

In reality, perturbations occur at each bounce due to factors such as surface asperities. Consequently, the velocity of the ball has both horizontal and vertical components. Letting d_1 and d_2 be defined by the horizontal distances between bounces 1 and 2 and bounces 2 and 3, respectively, then e would actually be derived from the ratio in Equation 2 as:

$$e = \frac{v_2}{v_1} = \frac{\sqrt{v_{2_{\text{horizontal}}}^2 + v_{2_{\text{vertical}}}^2}}{\sqrt{v_{1_{\text{horizontal}}}^2 + v_{1_{\text{vertical}}}^2}} = \sqrt{\frac{(d_2/t_2)^2 + g^2(t_2/2)^2}{(d_1/t_1)^2 + g^2(t_1/2)^2}} \tag{6}$$

However, since we capture positional information in the last three features of X, we decided to separate the temporal information from the positional information. Thus, we chose the simpler ratio of e for X_t, which only depended on t_1 and t_2.

B.2 Extracting X_p from ball bounce locations

The last three features of X, or X_p, encode positional information regarding the first three bounces. Letting $\vec{p}_1$ be the positional vector from the location of bounce 1 to the location of bounce 2, and $\vec{p}_2$ be the positional vector from the location of bounce 2 to the location of bounce 3 (see Figure 9), then the distances between the bounces can be calculated as:

$$d_1 = ||\vec{p}_1||_2 \text{ and } d_2 = ||\vec{p}_2||_2 \tag{7}$$

We use d_1 and d_2 as the second and third features of X. Combined with the temporal information in X_t, a more accurate estimation of e can be inferred.

The last feature of X_p encodes directional perturbations upon collision. Specifically, we find the signed angle α between $\vec{p}_1$ and $\vec{p}_2$ via the following expression:

$$\alpha = \text{sign}(\frac{\vec{p}_{1_x}\vec{p}_{2_y} - \vec{p}_{2_x}\vec{p}_{1_y}}{||\vec{p}_1||||\vec{p}_2||}) * \arccos(\frac{\vec{p}_1}{||\vec{p}_1||} \cdot \frac{\vec{p}_2}{||\vec{p}_2||}) \tag{8}$$

C Sound localization

In this section, we derive the Time Delay Estimation equations for 2D-localization using 3 microphones and compare the accuracy of offline and real-time sound localization methods.

C.1 Time Delay Estimation with 3 Microphones

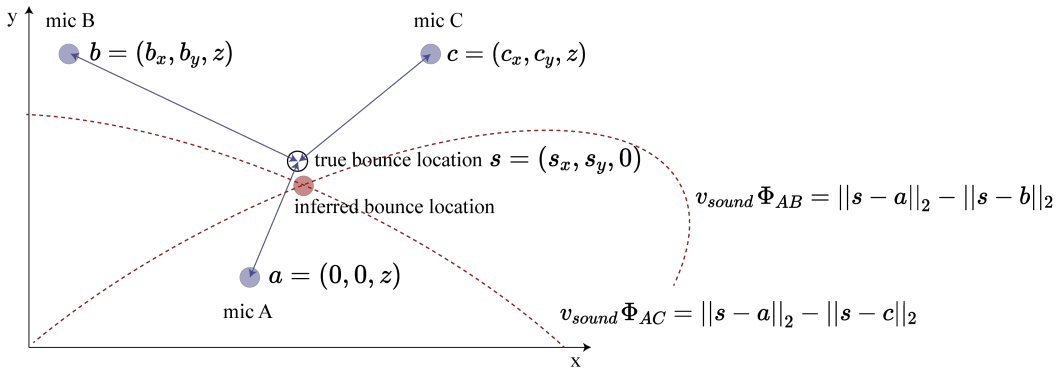


Figure 10: Locating the source of a sound using time delay estimation involves finding the intersection of two conics, defined by the time delays between pairs of microphones.

Let v_{sound} be the speed of sound through air, Φ_{AB} and Φ_{AC} be the time delays between signals received at microphones A and B and A and C, and denote the positions of the ball strike and microphones as $s = (s_x, s_y, 0)$, $a = (0, 0, z)$, $b = (b_x, b_y, z)$, and $c = (c_x, c_y, z)$, respectively. Then the time delay Φ_{AB} is a function of the relative distances of the microphones to the ball strike, or

$$\Phi_{AB} = \frac{1}{v_{sound}}(||s-a||_2 - ||s-b||_2) \tag{9}$$

Thus, solving for the location of the ball strike is equivalent to finding the intersection of two conics (see Figure 10):

$$v_{sound}\Phi_{AB} = ||s-a||_2 - ||s-b||_2 \text{ and } v_{sound}\Phi_{AC} = ||s-a||_2 - ||s-c||_2 \tag{10}$$

A nonlinear optimizer is used to find this intersection. Because there can be more than one solution, we initialize the optimizer with a location inside the convex hull of the microphone positions.

C.2 Comparison of Offline and Real-Time Sound Localization

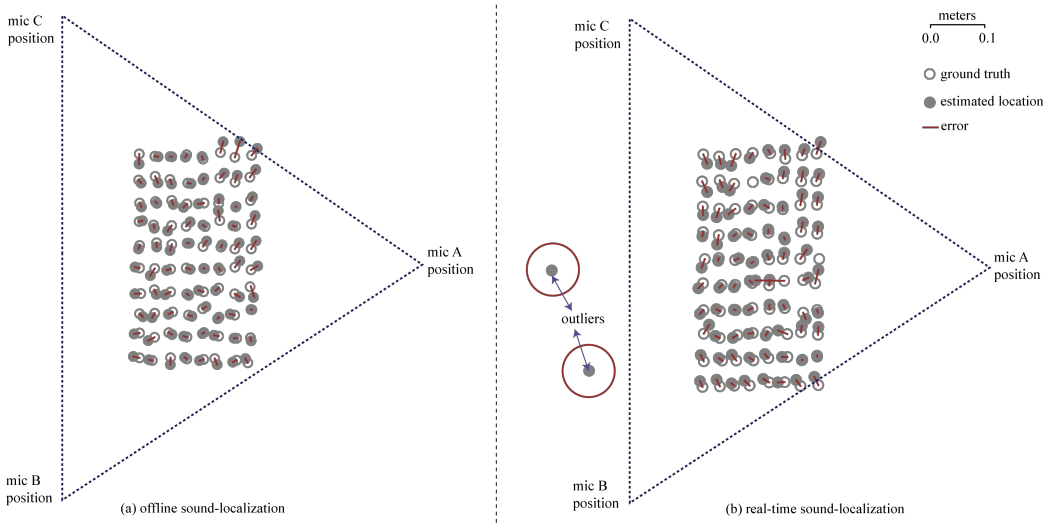


Figure 11: A comparison of (a) offline and (b) online sound localization. Ground truth positions were recorded by tapping a piece of paper layered with a carbon sheet. These positions were converted to world coordinates via an automated image processing script. Observe that outliers occasionally arise when using the less-accurate real-time sound localization method. However, the displacement errors of the outliers are sufficiently large such that they can be easily filtered.

D Dynamic Prediction without Collision Noise

Below, we supplement the description of dynamic predictions without collision noise in Section 3.4.1 with relevant equations. The robot control sequence is summarized in Algorithm 1.

Let (x_i, y_i), t_i, and d_i be the position of the most recent bounce, time, and distance between the most recent and prior bounces (see Figure 12). Assuming that the ball travels in a piecewise parabolic trajectory with no directional perturbations upon each bounce, we can estimate the exit velocity of the ball rebounding from the bounce to be defined by:

$$v_{e(xy_i)} = \frac{ed_i}{t_i} \text{ and } v_{e(z_i)} = \frac{egt_i}{2} \tag{11}$$

where g is the gravitational constant, the xy subscript refers to the horizontal velocity of the ball, and the z subscript refers to the vertical velocity component of the ball. These estimates allow us to calculate whether or not the ball will intersect the plane before the next bounce. First, denote d_R as the horizontal distance from the most recent bounce to the robot plane along the direction of travel (see Figure 12). The distance and time of the next bounce relative to the most recent bounce, or d_{i+1} and t_{i+1}, can be calculated by:

$$d_{i+1} = e^2 d_i \text{ and } t_{i+1} = et_i \tag{12}$$

Then to calculate whether or not the ball will intersect the robot plane before the next bounce, we check if

$$d_{i+1} > d_R \tag{13}$$

If true, the ball will intersect the plane before it lands again in t_R seconds, so the end-effector must be moved to the intersection point (x_R, y_R, z_R) in t_R seconds. If the ball will not intersect the plane before it lands again, the end-effector moves to y_R in t_{i+1} seconds before the next bounce is observed.

The intersection point (x_R, y_R, z_R) is partially pre-determined and therefore fixed by the placement of the robot (which sits at $x = x_R$). We can calculate y_R by finding the intersection between line

$x = x_R$ and the line connecting the last two ball bounce positions (see Figure 12). This intersection occurs at:

$$y_R = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} x_R + (y_i - \frac{y_i - y_{i-1}}{x_i - x_{i-1}} x_i) \tag{14}$$

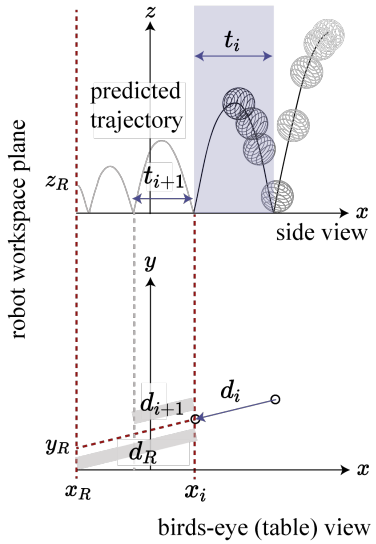


Figure 12: Dynamic predictions without collision noise assume a piecewise parabolic trajectory for the ball, based on the most recent two bounces. The robot moves to the intersection of this predicted trajectory and the robot workspace plane.

Finally, if it is determined that the ball will intersect the robot workspace plane before it bounces again, classical projectile equations can be used to calculate t_R and z_R:

$$t_R = \frac{d_R}{v_{e_{xy_i}}} = \frac{d_R t_i}{e d_i} \tag{15}$$

$$z_R = v_{e(z_i)} t_R - \frac{1}{2} g t_R^2 = (\frac{e g t_i}{2})(\frac{d_R t_i}{e d_i}) - \frac{1}{2} g (\frac{d_R t_i}{e d_i})^2 \tag{16}$$

Algorithm 1: Robot Control using Dynamic Predictions without Collision Noise

Result: Positions the end-effector of a robotic arm to meet a bouncing ball

while *Received bounce position and times* **do**
  $(x, y) \leftarrow$ bounce location;
  if *First bounce* **then**
    Move end-effector to position y;
  else
    if *After next bounce, the ball will intersect the robot plane at* (y_R, z_R) **then**
      Move end-effector to position (y_R, z_R) in t_R seconds;
    else
      Move end-effector to y_R in t_{i+1} seconds (before the next bounce is observed);
    end
  end
end

E Dynamic Predictions with Collision Noise

In this section, the algorithm for making dynamic predictions with collision noise is outlined in Algorithm 2. Additionally, a complete breakdown of all 30 tosses per ball is included, and the failure modes due to outliers are compared between Algorithm 1 and Algorithm 2. Finally, two real examples are shown with freeze-frames in Section E.3 to detail the algorithmic steps associated with each bounce event.

Algorithm 2: Robot Control using Dynamic Predictions with Collision Noise

Result: Positions the end-effector of a robotic arm to meet a bouncing ball

1 **while** *Received bounce position and times* **do**
2 $(x, y) \leftarrow$ bounce location;
3 **if** *First bounce* **then**
4 Move end-effector to position y;
5 **else if** *Second bounce* **then**
6 next_bounce_posterior $\leftarrow$ Generate posterior for next bounce position and time ;
7 Look ahead k bounces to generate n samples where the ball will intersect the plane ;
8 **if** *Any samples intersect the plane of the robot* **then**
9 Move $\propto \sigma^{-1}$ distance toward mean of samples, where σ is the standard deviation of the intersection points on the robot plane ;
10 **else**
11 **if** *Bounce position is an outlier* **then**
12 Replace bounce location with the mean of next_bounce_posterior ;
13 Repeat lines 6-9 ;
14 **end**
15 **end**

As discussed in Section 3.4.2, a new BayesSim model is trained to generate posteriors over the next bounce time and location, given the time and distance between the prior two bounces. After the second bounce is observed, this new BayesSim model can be used to generate a posterior for the next bounce position and time. Furthermore, the model can be used to recursively sample future bounce positions until they pass the robot workspace plane. For example, let k be the number of bounces to look ahead and n be the number of samples to generate at each future bounce. Given the last two bounces i and $i-1$, a posterior over the future bounce $i+1$ is generated using the BayesSim model. This posterior is sampled n times, and each sample can be used to create a new posterior with the most recent bounce i. Thus, n posteriors over bounce $i+1$ are generated. This is repeated up to k times until the sample intersects the workspace plane. A maximum of n^k samples are generated in this step. In experiments, we found that $n = 10$ and $k = [2, 4]$ were sufficient at capturing the uncertainty in future bounce locations and did not significantly add to the latency of the entire algorithm (this inference step ran at about 20 Hz). We varied k depending on how reactive we wanted the robot to be. For instance, the moonball is highly uncertain at each collision, so we used $k = 2$. In contrast, since the ping-pong ball experienced lower collision noise, we used $k = 4$.

At every subsequent bounce after the second bounce, the bounce location is classified as either an outlier or an inlier by using an elliptic envelope filter with the future bounce posterior generated at the prior bounce. If the bounce location is determined to be an outlier, then the mean of that posterior is used in place of the outlier location. This extra check was implemented to compensate for inaccurate real-time bounce localization due to high reverberation. In the following section, we compare the sensitivity to outliers using Algorithms 1 and 2.

E.1 Full Breakdown of Success and Failures out of all 30 Tosses

Section 4.4 performs analysis on the tosses that do not violate the robot's workspace constraints. For completeness, we include a full breakdown of all 30 tosses in Figure 13, which includes the failures due to the ball violating the robot workspace boundaries. As shown in Figure 13, few to none of the tosses violate the boundary for the tennis ball and ping pong ball, but almost a third of the tosses for the moon ball result in boundary violations. This is because the moonball experiences highly stochastic collisions.

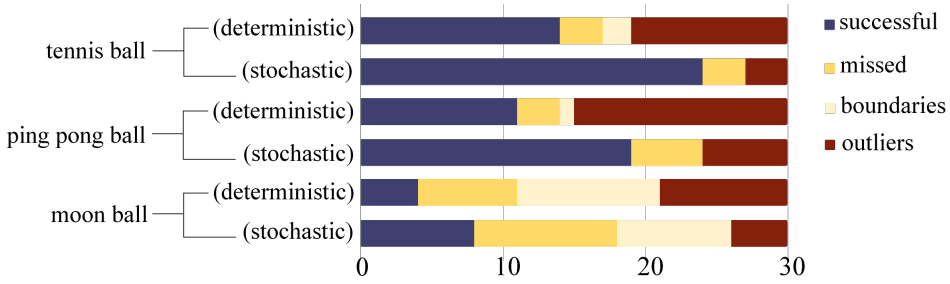


Figure 13: "Outliers" signifies failures due to errors in sound localization, "Boundaries" refers to failures due to the ball violating robot workspace boundaries, and "Missed" denotes any other failure. All 30 tosses are represented in this graph.

E.2 Comparison of Sensitivity to Outliers with Algorithms 1 and 2

In Figure 14, we break down the failure modes of both Algorithms 1 and 2 due to outliers in sound localization. As shown in the graph, both methods are sensitive to outliers in the first or second bounce. Because Algorithm 2 relies on accurate first and second bounce locations to predict future bounce distributions, there currently is not a way to adjust for these errors. We believe that we can eliminate these outliers by implementing more efficient real-time processing, for instance, by using a real-time audio-processing language called SuperCollider, or by training a neural network to perform fast peak detection. However, by considering collision noise for dynamic predictions, Figure 14 shows that Algorithm 2 successfully filters out outliers in any subsequent bounces and can compensate for their errors.

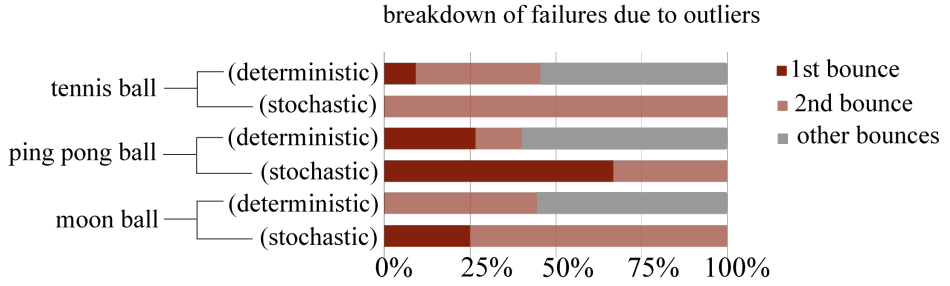


Figure 14: We categorize the failures due to outliers in bounce localization by the bounce at which the outlier occurs. Note that Algorithm 2 successfully filters out outliers as long as they occur after the second bounce. The robot is able to use, instead, the mean of the posterior over which the bounce *should* have occurred, which subsequently leads to more accurate ball trajectory predictions.

E.3 Example robot trials with captions

Below, we detail two real robotic examples with freeze-frames to depict the steps of Algorithm 2.

E.3.1 Example 1: Success with no outliers

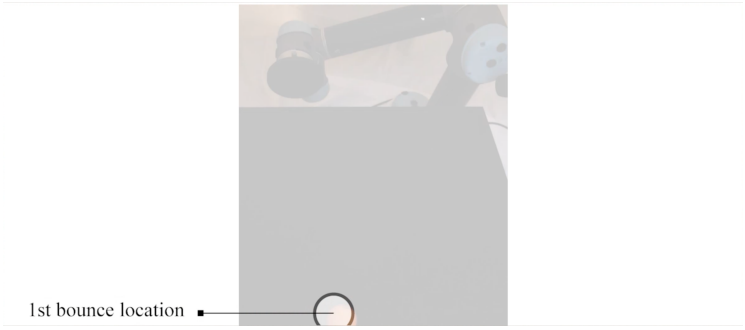


Figure 15: The first bounce is located.

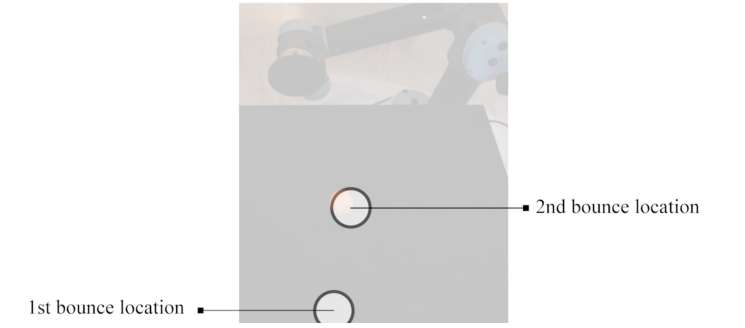


Figure 16: The second bounce is located.

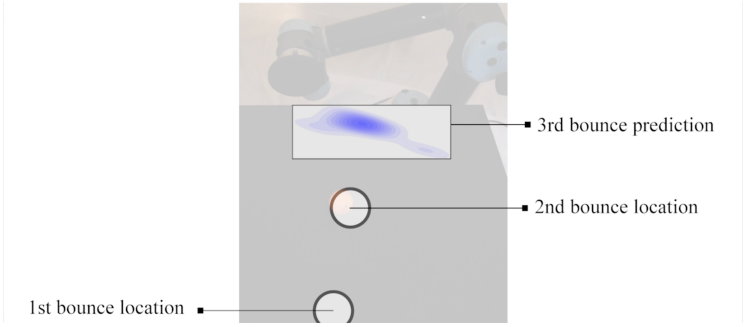


Figure 17: The 1st and 2nd measured locations are used to predict the distribution over the 3rd bounce location.

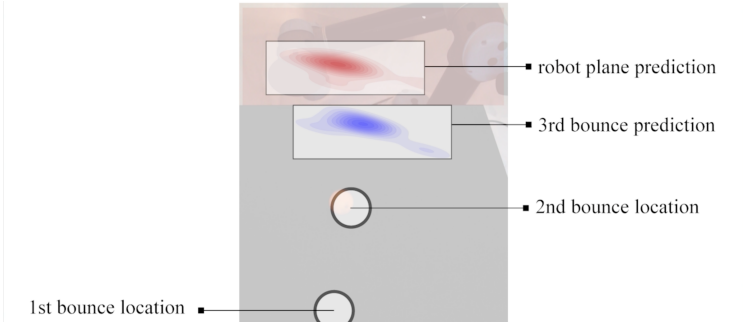


Figure 18: The 2nd measured location and the 3rd bounce location distribution are used to predict the distribution of the robot plane intersections.

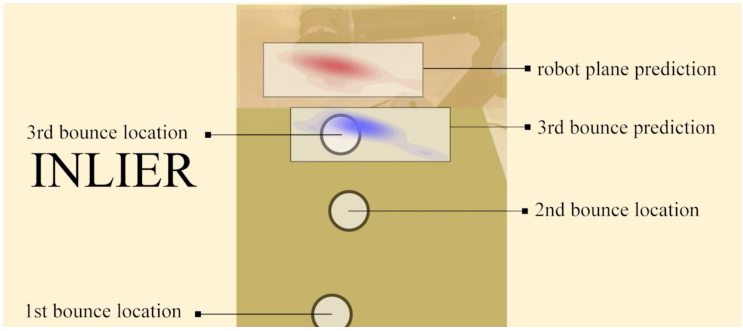


Figure 19: Inlier: the audio-based localization of the 3rd bounce lies within the predicted distribution and is trusted. This leads to a successful prediction of the robot plane intersection, and the robot comes into contact with the ball

E.3.2 Example 2: Success with an outlier

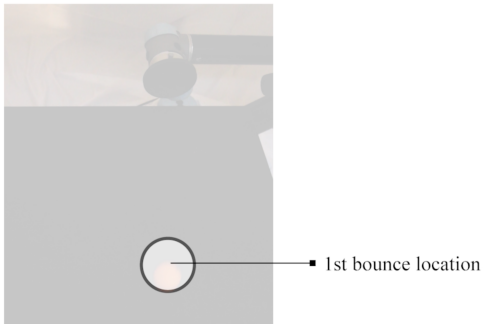


Figure 20: The first bounce is located.

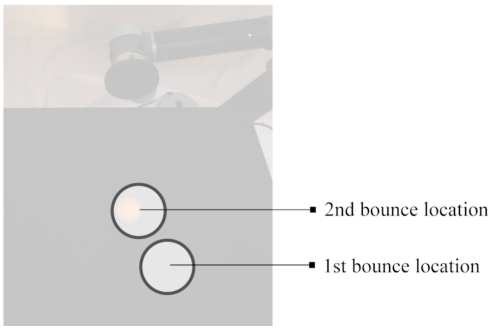


Figure 21: The second bounce is located.

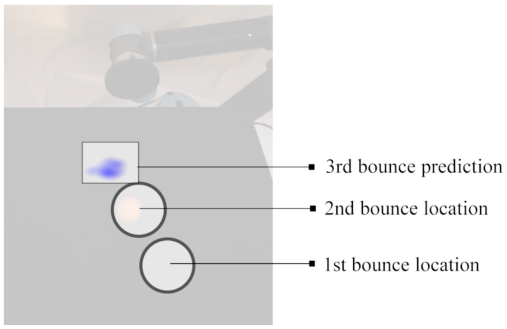


Figure 22: The 1st and 2nd measured locations are used to predict the distribution over the 3rd bounce location.

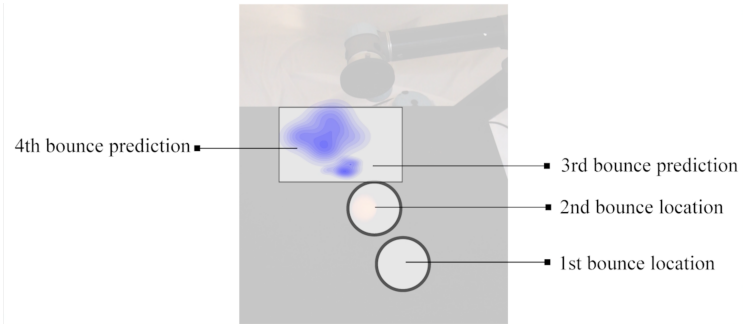


Figure 23: The 2nd measured location and 3rd bounce location distribution are used to predict the distribution over the 4th bounce location.

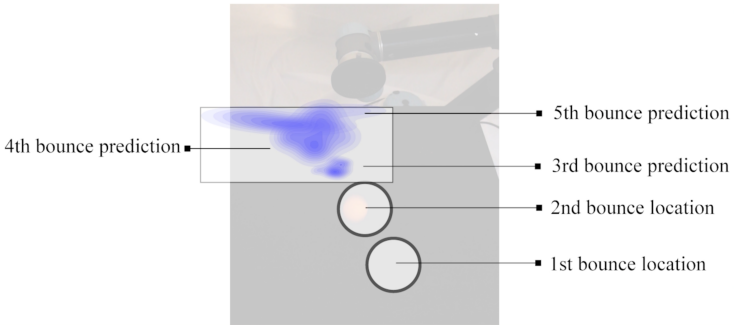


Figure 24: The 3rd and 4th bounce location distributions are used to predict the distribution over the 5th bounce location.

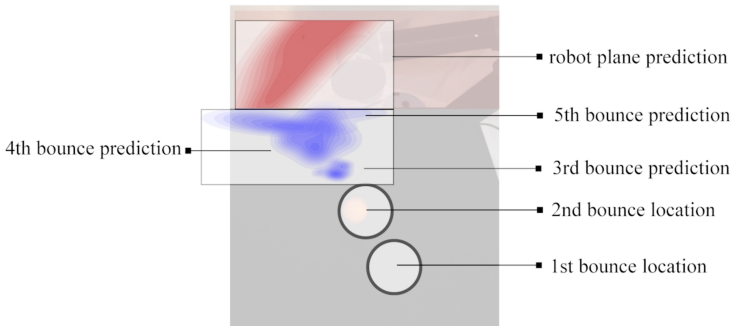


Figure 25: The 4th and 5th bounce location distributions are used to predict the distribution of the robot plane intersections.

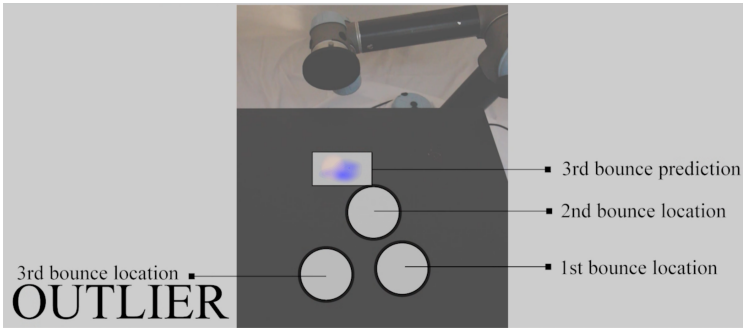


Figure 26: Outlier: the audio-based localization of the 3rd bounce lies far from the predicted distribution and is ignored.

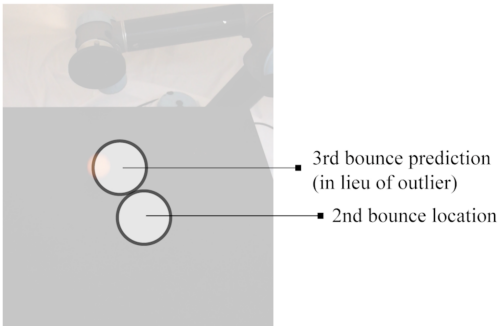


Figure 27: The mean of the predicted distribution for the 3rd bounce is used in lieu of the outlier.

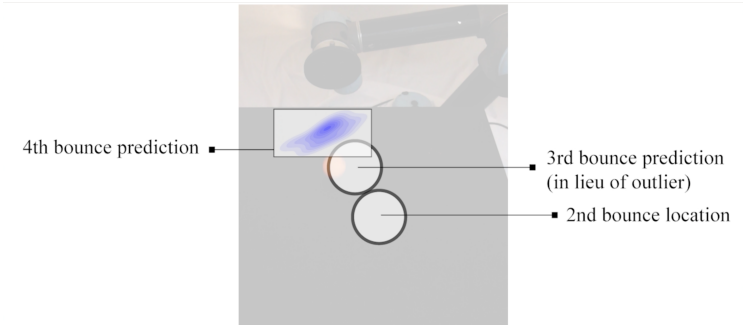


Figure 28: The 2nd measured location, as well as the 3rd *predicted* location, are used to predict the distribution of the 4th bounce.

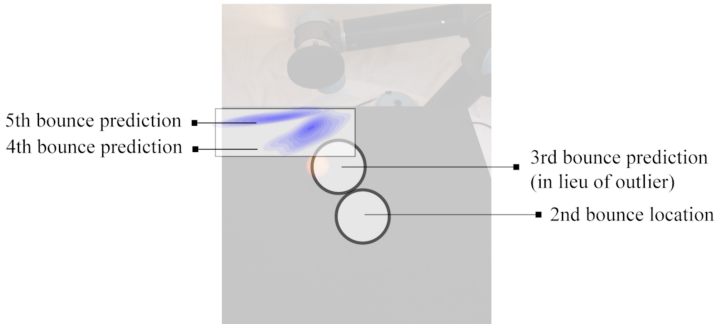


Figure 29: The 3rd *predicted* location, as well as the 4th bounce location distribution, are used to predict the distribution of the 5th bounce.

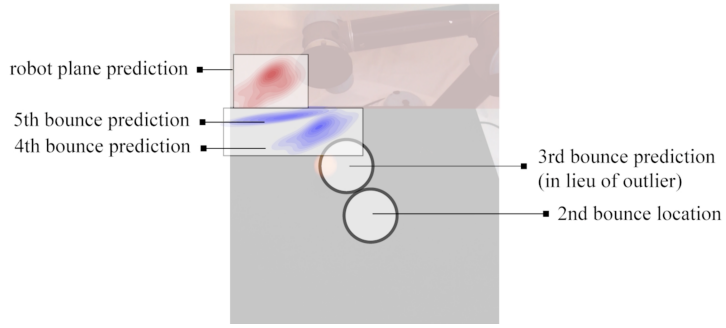


Figure 30: The 4th and 5th bounce location distributions are used to predict the distribution of the robot plane intersections.

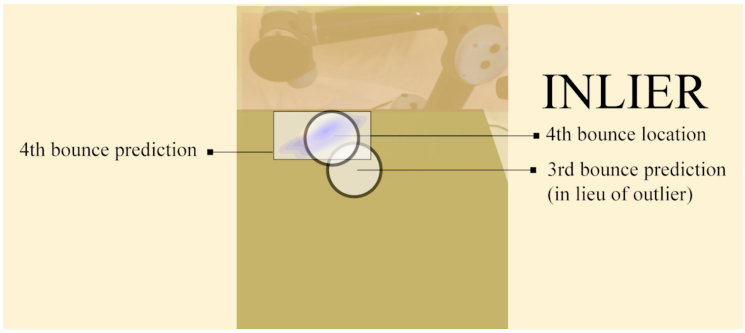


Figure 31: Inlier: the audio-based localization of the 4th bounce lies within the predicted distribution.

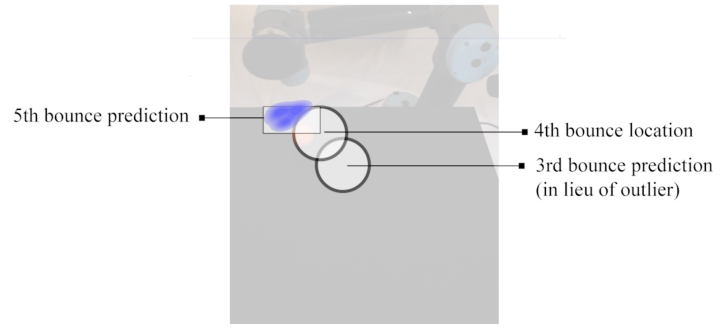


Figure 32: The 3rd *predicted* location, as well as the 4th measured location, are used to predict the distribution of the 5th bounce.

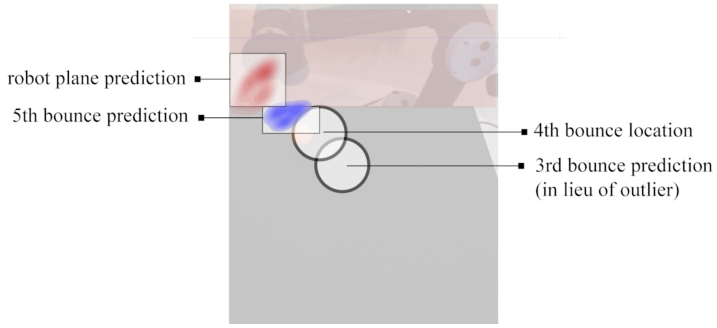


Figure 33: The 4th measured location and the distribution of the 5th bounce are used to predict the distribution of the robot plane intersections.

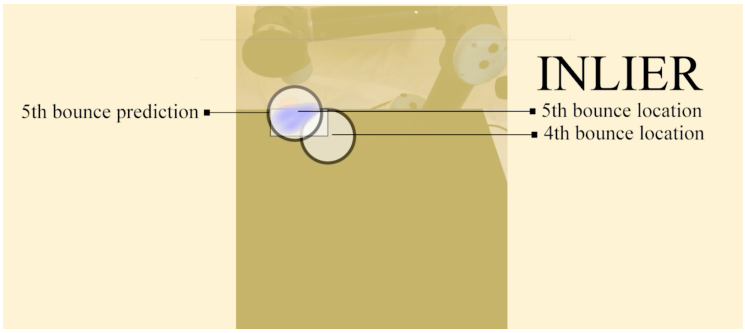


Figure 34: Inlier: the audio-based localization of the 5th bounce lies within the predicted distribution.

Figure 35: Successful prediction