PhysicsGen – Supplementary Material

This Appendix provides additional details and supplementary material supporting the main content of the paper. Below are the sections included in this Appendix:

A. Training Setup

This appendix provides a detailed overview of the architecture, input specifications, hyperparameters, and computer resources used for the generative models utilized in this research. Each experiment was conducted on a workstation equipped with a single NVIDIA RTX 4090 GPU (24 GB VRAM) unless specified otherwise.

The U-Net model architecture, adopted from [31], and the Pix2Pix setup, based on [20], are designed to process either grayscale or RGB image inputs. When necessary, the input is extended by appending additional parameters as a separate dimension. We followed the methodology described in [18] for the Diffusion model while incorporating conditional inputs as separate dimensions alongside the noised input image. Each model was constructed upon a unified U-Net backbone, scaling from 64 to 1028 channels and reconverging to 64, ensuring consistency in model design across all datasets. During training, different loss functions were employed to suit each model's architecture: Mean Squared Error (MSE) loss for the U-Net and Diffusion models, and a combination of Binary Cross Entropy (BCE) loss and L1 loss for the GAN. These architectures were applied consistently across all tasks introduced in this paper, with each task trained and evaluated independently. The ConvAE and VAE [25] models are both based on the U-Net architecture, similar to Pix2Pix and the standard U-Net, but adapted for their respective tasks. The ConvAE uses the U-Net structure with skip connections disabled. The VAE employs a similar hierarchical design.

The Stable Diffusion models were trained following the training procedure outlined in [30], with variations in conditioning mechanisms. In the standard Stable Diffusion (SD) setup, the conditioned image and parameters were passed to the denoising model via cross-attention and as additional dimensions with the noised input image. In contrast, the Stable Diffusion with Cross Attention Only (SD wCA) model exclusively employed cross-attention for passing the conditioned image and parameters. The DDBMs were trained following the exact training procedure outlined in [38], ensuring adherence to the methods described in the original work. DDBMs were trained on four NVIDIA A100 GPUs (80 GB VRAM each) to accommodate the increased computational demand, while Stable Diffusion was trained on a single A100 GPU. All other models, including U-Net, Pix2Pix, and standard Diffusion models, were trained on a single NVIDIA RTX 4090 GPU.

Table 4. UNet Training Hyperparameter

Hyperparameter	Value
Batchsize	18
Learning Rate	1×10^{-4}
Epochs	50
Optimizer	Adam
Adam Betas	(0.5, 0.999)

Table 5. Pix2Pix Training Hyperparameter

Hyperparameter	Value
Batchsize	18
Learning Rate Discriminator	1×10^{-4}
Learning Rate Generator	2×10^{-4}
Epochs	50
L1 Lambda	100
Lambda GP	10
Optimizer	Adam
Adam Betas	(0.5, 0.999)

Table 6. DDPM Training Hyperparameter

Hyperparameter	Value
Batchsize	18
Learning Rate	1×10^{-4}
Epochs	50
Noise Steps	1000
Optimizer	Adam
Adam Betas	(0.5, 0.999)

Table 7. StableDiffusion Training Hyperparameters

Hyperparameter	Value
Number of Timesteps	1000
Beta Start	0.0015
Beta End	0.0195
Down Channels	[256, 384, 512, 768]
Mid Channels	[768, 512]
Number of Heads	16
LDM Batch Size	16
Autoencoder Batch Size	4
Discriminator Start	5000
LDM Learning Rate	1×10^{-4}
Autoencoder Learning Rate	1×10^{-5}
Codebook Weight	1
Commitment Beta	0.2
Perceptual Weight	1
KL Weight	0.000005
LDM Epochs	100

Table 8. DDBM Training Hyperparameters

Hyperparameter	Value
σ_{max}	80.0
σ_{min}	0.002
σ_{data}	0.5
Covariance (C_{XY})	0
Number of Channels	256
Attention Levels	[32, 16, 8]
Number of Residual Blocks	2
Sampler	real-uniform
Attention Type	flash
Learning Rate	0.0001
Dropout	0.1
EMA Rate	0.9999
Number of Head Channels	64

B. Sound Propagation Appendix

B.1. Evaluation Metrics

This appendix section details the evaluation metrics used, focusing on the Weighted Mean Absolute Percentage Error (wMAPE) and the implementation of ray tracing to determine line-of-sight (LoS) conditions between a sound source and various points on a sound propagation map.

Weighted Mean Absolute Percentage Error: The calculation of the wMAPE is adjusted to address cases where the true value of the noise map is zero but the predicted value is non-zero. In such instances, the script sets the error for these specific pixels to 100%. This method accurately quantifies the total error, especially when the model predicts sound in areas where sound waves could not realistically reach according to the true data.

Line of Sight: Ray tracing is employed to establish LoS conditions between a sound source and various points on a sound propagation map. The process initiates by establishing a grid that represents the mapped area, with the sound source positioned at the center. For each point on this grid, a direct line is drawn from the source to the point. The script then checks for any obstructions along this line.

The ray-tracing function progresses by incrementally moving along the line from the source to the target point. It checks if any part of the line intersects with obstacles, represented by zero values on a binary image map. If an obstruction is encountered before the line reaches the target point, that point is marked as not having a line of sight to the source; otherwise, it is considered to have a line of sight.

B.2. Location Sampling

In order to conduct sound propagation studies that are reflective of diverse urban environments, our location sampling was designed with specific criteria to ensure a balanced and comprehensive dataset. The location sampling methodology for sound propagation studies required each selected location to contain at least ten buildings within a 200-meter radius of the designated sound source. Additionally, to avoid anomalous acoustic results and to simulate a realistic setting where sound sources are typically not positioned directly against structures, no buildings were allowed within a 50-meter radius of the sound source. Figure 9 illustrates the urban areas selected for this study, showcasing the distribution of buildings relative to the sound source in the center.

Locations were randomly sampled across ten cities, providing a broad geographical spread and a variety of urban layouts. The locations sampled include: Hamburg, Hannover, Augsburg, Bonn, Munich, Schwerin, Berlin, Paris, Stuttgart, and Aachen.

For each city, 2500 unique locations were selected, contributing to a total of 25,000 data points for the study. The dataset was then divided into training, evaluation, and testing sets with a split of 80%, 15%, and 5%.

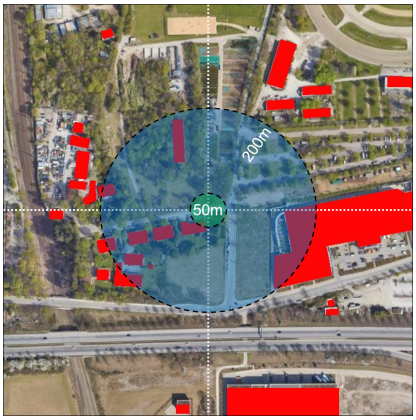


Figure 9. Satellite image of a sampled location for sound propagation studies. Buildings are marked in red, illustrating the distribution within the area. The blue circle represents the 200-meter radius within which at least ten buildings are required, while the green circle indicates the 50-meter radius that must be free of any buildings to ensure a clear propagation path from the sound source.

B.3. Receiver Placement

The receiver placement for our sound propagation simulations is a critical component that directly influences the accuracy and relevance of our results. To achieve an optimal setup, we utilized the NoiseModelling framework [4]. Our specific placement strategy required a combination of precision and broad coverage, which was not fully supported by any single existing script within the framework. Therefore, we integrated two available scripts and further developed a custom WPS (Web Processing Service) to meet our specific needs.

The first script we used was **Regular_Grid**, which calculates a regular grid of receivers. This script uses a single geometry or a table of geometries to generate receivers evenly spaced by a specified distance (*delta*) on the Cartesian plane, measured in meters. This method ensures a systematic and uniform coverage across the studied area, providing a comprehensive baseline for sound propagation assessments.

(a) Sattelite image (b) Receiver grid (c) Urban layout

Figure 10. Starting with the selection of a 500m² area (a), buildings are identified, followed by placing a receiver grid (b). The urban layout (c) is then used for creating samples for the dataset.

The second script, **Building_Grid**, is designed to place receivers specifically around building facades. This script generates receiver points approximately 2 meters from the building facades at a given height, facilitating detailed analysis of sound interactions with building surfaces, which are critical for urban acoustic modeling.

By combining these two approaches, we crafted a receiver placement strategy that not only maintains a uniform grid for broad coverage but also includes strategically placed receivers around building facades and edges. This hybrid approach ensures that our simulations accurately reflect sound propagation both across open areas and in close proximity to structural barriers, thereby enhancing the precision of the simulation results at key locations.

B.4. Sound Physics

Following [35], for a discrete set of receivers R, the amplitude $L^j_{R_k}$ of receiver R_k at frequency j is computed via iterative differences:

$$L^j_{R_k} = L^j_W - A_{div_{R_k}} - A^j_{atm_{R_k}} - A^j_{dif_{R_k}} - A^j_{grd_{R_k}} \tag{8}$$

where

L^j_W models the source,
$A_{div_{R_k}}$ captures the geometrical spreading,
$A^j_{atm_{R_k}}$ represents the atmospheric absorption,
$A^j_{dif_{R_k}}$ models diffraction,
$A^j_{grd_{R_k}}$ the ground effect - which is neglected in our study.

Several environmental factors influence the sound level at a receiver:

Geometrical Spreading: This factor accounts for the dispersion of sound waves as they propagate through the medium. The decrease in sound intensity due to geometrical spreading is given by:

$$A_{div_{R_k}} = 20\log_{10}(d_i) + 11 \tag{9}$$

where d_i is the distance between the sound source and the receiver.

Atmospheric Absorption: This factor represents the loss of sound energy as it travels through the atmosphere, influenced by the atmospheric conditions such as humidity and temperature. It is calculated as follows:

$$A^j_{atm_{R_k}} = \alpha_{air}\frac{d_i}{1000} \tag{10}$$

where α_{air} is the atmospheric absorption coefficient.

Table 9. Model vs. Simulation Performance Comparison for Single Sample Processing. The complex source is a single test sample for a more complex source with 28 descriptive sound signal sources for the simulation. This illustrates how the processing time increases significantly with more complex signal sources. It is important to note that this analysis may not provide a completely fair assessment from a theoretical perspective, as no efforts were made to optimize the simulation codes for GPU execution.

Model - Condition	**Mean Runtime (ms)**
convAE	0.128
UNet	0.126
Pix2Pix	0.138
DDPM	3986.353
SD(w.CA)	2961.027
SD	2970.86
DDBM	3732.21
Simulation - Baseline	20471.7
Simulation - Diffraction	20602.7
Simulation - Reflection	25097.3
Simulation - Combined	29239.5
Simulation - Combined - 3rd Order Reflections	186229.5
Simulation - Combined - Complex Source	540000

Diffraction: Sound waves can bend around obstacles, a phenomenon modeled by the diffraction factor:

$$A^j_{dif_{R_k}} = \begin{cases} 10\log_{10}(3 + \frac{40}{\lambda}C''\delta) & \text{if } \frac{40}{\lambda}C''\delta \geq -2 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where λ is the wavelength of the sound, C'' is the diffraction coefficient, and δ is the path length difference expressed in m between for direct and diffracted paths.

Reflection Adjustments: Reflections off various surfaces can significantly modify the sound level. This factor adjusts the sound power level based on the number of reflections and the properties of the reflective surfaces:

$$L_W^{(n_{ref})} = L_W^{(n_{ref}-1)} + n_{ref} \times 10\log_{10}(1 - \alpha_{vert}) \tag{12}$$

where n_{ref} is the number of reflections and α_{vert} is the absorption coefficient of the reflecting surfaces.

B.5. Runtime Analysis

The runtime analysis, as detailed in Table 9, highlights the performance comparison between various models and simulation approaches for single sample processing.

B.6. Scalable Simulation Pipeline

The dataset generation pipeline visualized in Figure 11 is a crucial component of our study, developed to efficiently process sound propagation data in diverse urban settings.

Utilizing the NoiseModelling framework [4], we have automated the data input and simulation processes within a Docker-containerized environment. The pipeline commences with the automatic download of a 500m² area map from OpenStreetMap for each location using the Overpass API, followed by their import into the NoiseModelling framework alongside the signal source.

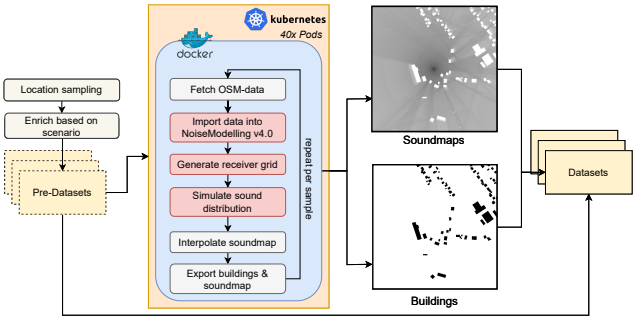


Figure 11. Detailed visualization of the dataset generation pipeline.

Considering the computational intensity of this process, with an average duration of 30 seconds per sample, our pipeline is structured for scalability. It operates on a Kubernetes cluster with 40 pods, enabling us to complete the generation of the entire dataset, encompassing 25,000 data points for each complexity level, in approximately 20 hours.

B.7. Additional Qualitative Results

Figure 13 visualizes the error patterns for Pix2Pix, Unet and DDPM, illustrating behavior that is similar across all the architectures evaluated. The Unet model tends to blur areas where it is uncertain about the sound propagation results. Instead of accurately replicating the complex reflection patterns found in the dataset, the GAN attempts to fill these uncertain areas with a simplified, repetitive texture that does not match the true reflection dynamics.

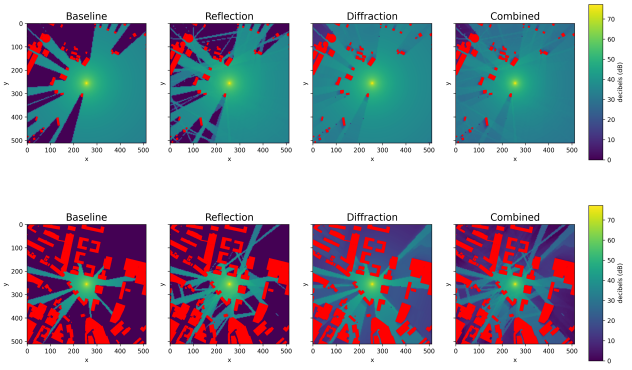


Figure 12. Comparison of true labels for **baseline**, **reflection**, **diffraction**, and **combined** tasks for two samples to visualize their differences.

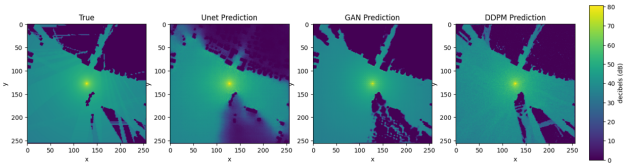


Figure 13. Visualization of unique error patterns for Pix2Pix, UNet, and DDPM models in sound propagation simulations. Each model's approach to uncertain areas and its replication of reflection patterns are depicted.

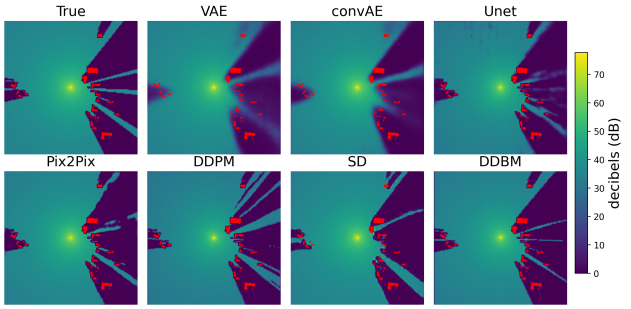


Figure 14. Comparing the output of the physical simulation with the predictions for a single sample within the **baseline** task dataset.

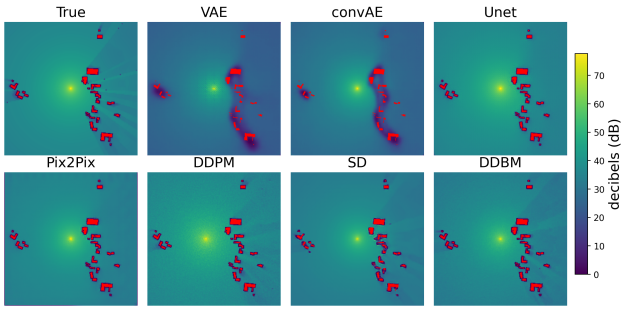


Figure 15. Comparing the output of the physical simulation with the predictions for a single sample within the **diffraction** task dataset.

In contrast, the diffusion model attempts a more direct replication of the reflection patterns visible in the training data. However, the error patterns observed with diffusion models, including DDPM, SD, and DDBMs, are highly similar. These models tend to overcompensate, introducing reflection patterns into areas where they are not physically plausible. This results in a noisy and sometimes unrealistic simulation output, particularly in regions where sound reflections should be minimal or absent based on the physical layout.

Table 10. Quantitative evaluation across all tasks for all architectures with a batch size of 16 during inference. Best overall results in **bold**, second best underlined.

Model	Task	MAE ↓		wMAPE ↓		Runtime/ ↓
		LoS	NLoS	LoS	NLoS	Sample (ms)
Simulation	Base	0.00	0.00	0.00	0.00	204700
convAE	Base	3.67	2.74	20.24	67.13	0.128
VAE [25]	Base	3.92	2.84	21.33	75.58	0.124
UNet [31]	Base	2.29	1.73	12.91	37.57	0.138
Pix2Pix [20]	Base	1.73	1.19	**9.36**	**6.75**	0.138
DDPM [18]	Base	2.42	3.26	15.57	51.08	3986.353
SD(w.CA) [30]	Base	3.76	3.34	17.42	35.18	2961.027
SD	Base	2.12	**1.08**	13.23	32.46	2970.86
DDBM [38]	Base	**1.61**	2.17	17.50	65.24	3732.21
Simulation	Dif.	0.00	0.00	0.00	0.00	206000
convAE	Dif.	3.59	8.04	13.77	32.09	0.128
VAE	Dif.	3.92	8.22	14.46	32.57	0.124
UNet	Dif.	0.94	**3.27**	4.22	22.36	0.138
Pix2Pix	Dif.	**0.91**	3.36	**3.51**	**18.06**	0.138
DDPM	Dif.	1.59	**3.27**	8.25	20.30	3986.353
SD(w.CA)	Dif.	2.46	7.72	10.14	31.23	2961.027
SD	Dif.	1.33	5.07	8.15	24.45	2970.86
DDBM	Dif.	1.35	3.35	11.22	23.56	3732.21
Simulation	Refl.	0.00	0.00	0.00	0.00	251000
convAE	Refl.	3.83	6.56	20.67	93.54	0.128
VAE	Refl.	4.15	6.32	21.57	92.47	0.124
UNet	Refl.	2.29	5.72	12.75	80.46	0.138
Pix2Pix	Refl.	2.14	**4.79**	**11.30**	**30.67**	0.138
DDPM	Refl.	2.74	7.93	17.85	80.38	3986.353
SD(w.CA)	Refl.	3.81	6.82	19.78	81.61	2961.027
SD	Refl.	2.53	5.26	15.04	55.27	2970.86
DDBM	Refl.	**1.93**	6.38	18.34	79.13	3732.21
Simulation	Comb.	0.00	0.00	0.00	0.00	251000
convAE	Comb.	2.93	5.19	18.61	48.92	0.128
VAE	Comb.	3.08	5.35	19.59	50.24	0.124
UNet	Comb.	1.39	2.63	10.10	45.15	0.138
Pix2Pix	Comb.	1.37	2.67	**9.80**	40.68	0.138
DDPM	Comb.	**1.26**	**2.21**	13.07	**40.38**	3986.353

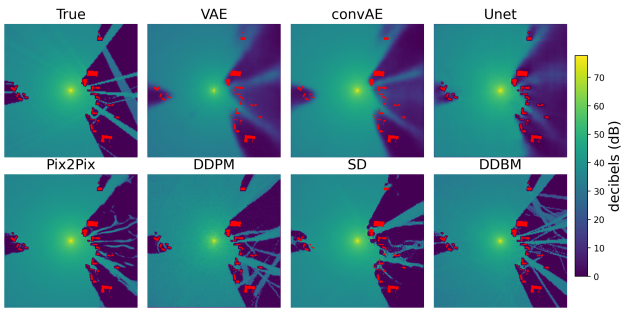


Figure 16. Comparing the output of the physical simulation with the predictions for a single sample within the **reflection** task dataset.

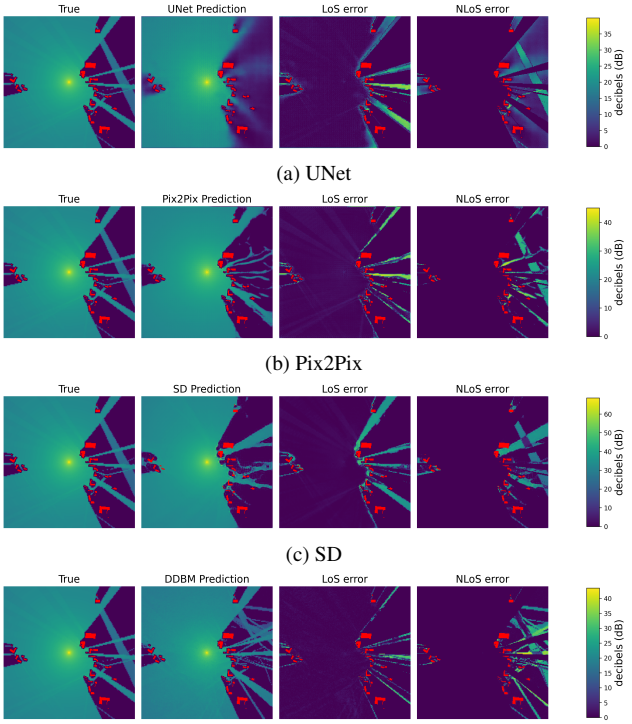


Figure 17. Comparing the output of the physical simulation with the prediction of UNet (a), Pix2Pix (b), stable diffusion model (c) and DDBM (d) for a single sample within the **reflection** task dataset, distinguishing between the MAE in LoS and NLoS.

C. Lens Appendix

C.1. Evaluation Metrics for Facial Landmark Detection

This appendix section details the evaluation metrics used to assess the accuracy of facial landmark detection, focusing on the Euclidean distance and mean absolute error calculations for both combined and separate x and y coordinates.

Euclidean Distance: The primary metric for evaluating the accuracy of facial landmark predictions is the Euclidean distance between predicted and true landmarks. This measure computes the root mean square error across all landmarks, providing a comprehensive gauge of overall prediction accuracy.

Mean Absolute Error for X and Y Coordinates: Additionally, the MAE for x and y coordinates is calculated separately. This breakdown allows for a detailed analysis of the model's performance along each axis, highlighting any directional biases or discrepancies in landmark prediction.

These metrics provide a dual approach to evaluating landmark detection performance—offering both a holistic measure via Euclidean distance and a directional sensitivity via separate MAE calculations.

C.2. Lens Physics

The Brown-Conrady model is used in image processing to correct lens distortions by modeling both radial and tangential components [12]. While the paper focuses on the tangential aspects of distortion, here we provide a detailed mathematical formulation of the entire Brown-Conrady model, including both radial and tangential distortion corrections.

Radial distortion is typically dominant in optical systems and is characterized by its impact varying with the square of the distance from the optical center. This effect is modeled using three coefficients: k_1, k_2, and k_3, which adjust the coordinates as a function of radial distance $r^2 = x^2 + y^2$. The full expressions for the distorted coordinates incorporating both radial and tangential distortions are as follows:

$$x_{dist} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{13}$$

$$+x + \left[2p_1 xy + p_2\left(r^2 + 2x^2\right)\right], \tag{14}$$

$$y_{dist} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{15}$$

$$+y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2 xy\right]. \tag{16}$$

These equations describe how each image point (x, y) is displaced to (x_{dist}, y_{dist}) due to lens distortions.

C.3. Lens Distortion Pipeline

This appendix details the methodology for applying lens distortions to images within our dataset, using a computational pipeline built on Python and OpenCV [5]. The process leverages a pre-defined set of distortion parameters stored in a CSV file, including tangential coefficients, to systematically alter each image.

Distortion Computation: Each image is processed sequentially. The script calculates the distortion for each pixel using the specified parameters (radial coefficients k_1, k_2, k_3 and tangential coefficients p_1, p_2) and the camera calibration data (focal lengths fx, fy and principal point coordinates cx, cy). These calculations transform the original pixel coordinates to their new distorted positions.

Image Remapping: Using OpenCV's `remap` function, the distorted pixel coordinates are mapped back onto the original image, creating the visually distorted output. This process ensures that each pixel's new location reflects the simulated lens.

C.4. Additional Qualitative Results

This section provides a visual evaluation of the generative models' performance on the lens distortion task, showcasing some examples of model predictions and landmark detection.

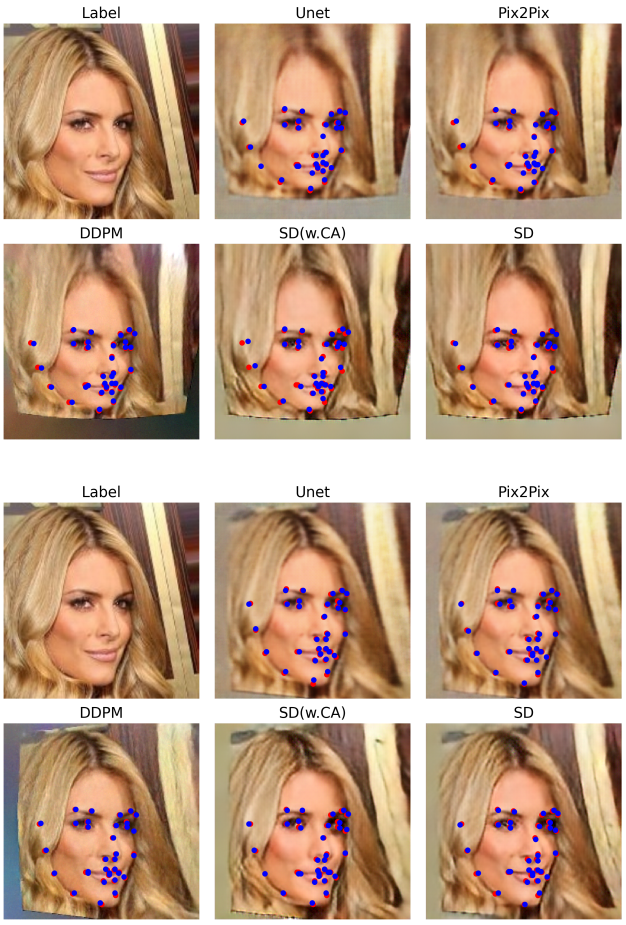


Figure 18. Comparison of simulation data with predicted landmarks for a single sample. The left image shows the input, while the right image overlays the predicted (blue dots) and true (red dots) landmark positions on the model output. For clarity, only every second landmark is visualized.

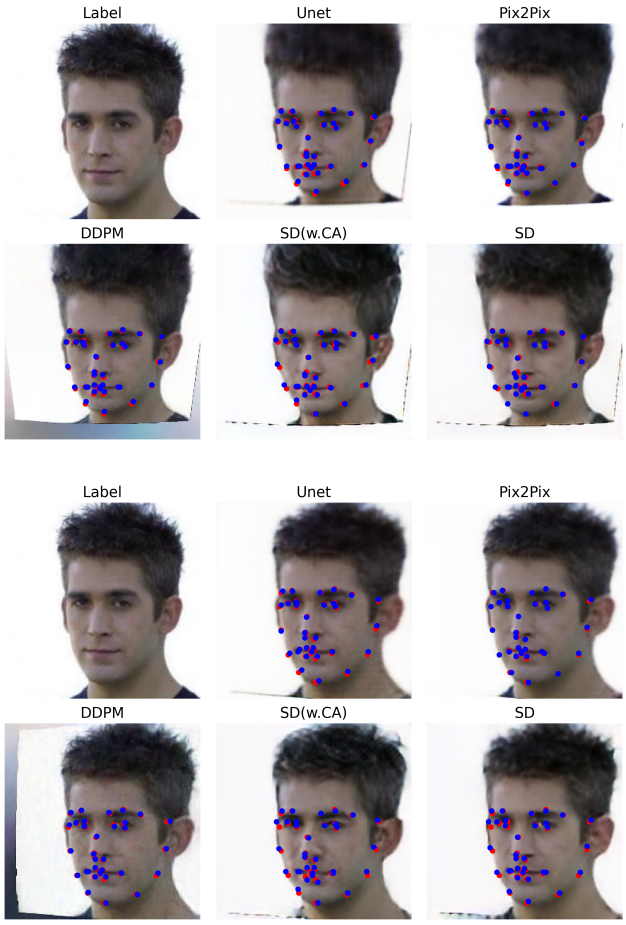


Figure 19. Comparison of simulation data with predicted landmarks for a single sample. The left image shows the input, while the right image overlays the predicted (blue dots) and true (red dots) landmark positions on the model output. For clarity, only every second landmark is visualized.

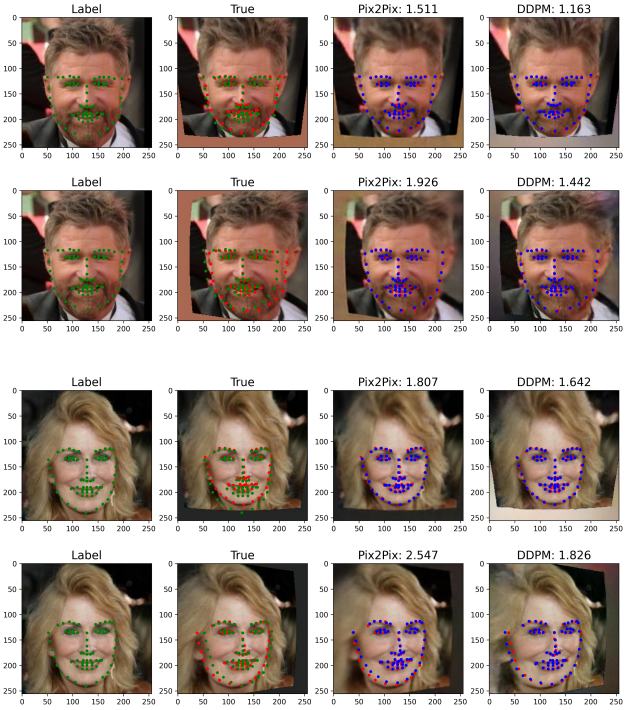


Figure 20. Comparing the input and output of the simulation with the predictions from Pix2Pix and DDPM for two random samples within the lens distortion task dataset. Green dots represent detected landmarks in the label image, red dots indicate the actual landmarks post-lens distortion, and blue dots denote the predicted landmarks by the models.

D. Ball Appendix

D.1. Ball physics and kinematics

Rolling ball. The kinematic problem of the rolling ball can be relatively simply described using Newton's law and the angular momentum balance equation.
The equations of motion for the X and Y directions are derived from the force balance acting on the ball, where no movement occurs in the Y direction (see figure 21):

$$\vec{F_{all}} = m * \vec{a} \quad \text{with} \quad \vec{F_{all}} = \vec{F_G} + \vec{F_N} + \vec{F_{HR}}$$

Equation of motion along x:

$$m * g * \sin(\beta) - F_{HR} = m * \ddot{x} \tag{17}$$

Equation of motion along y (no movement):

$$F_N - m * g * \cos(\beta) = 0 \tag{18}$$

To describe the rotation and the ball angle, the angular momentum balance from rotational mechanics can be used:

$$F_{HR} * r = J * \alpha > 0 \Leftrightarrow F_{HR} * r = J * \ddot{\varphi} > 0 \tag{19}$$

In the description of this first simple case, the ball position along the X-axis and the ball angle are derived twice.

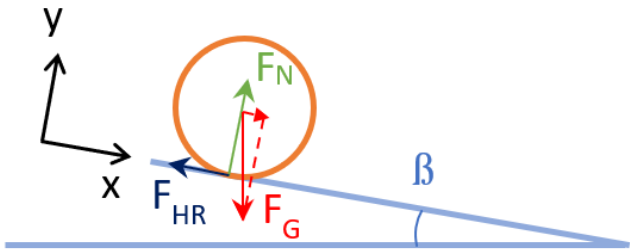


Figure 21. Forces overview for the rolling ball

Bouncing ball. The movement of the bouncing ball is much more complex to describe than the rolling case. For a better overview, the movement is divided into different sections (see different colors in figure 22).
First, the ball is released from a defined height without any initial velocity (green area). Until it hits the inclined ground surface, it is in free fall. Derived from Newton's law, the following equations applies:

$$F_g = m * a \Leftrightarrow m * \ddot{y} = -m * g \Leftrightarrow \ddot{y} = -g \tag{20}$$

$$\Rightarrow y(t) = y_0 - \frac{g}{2} * t^2 \tag{21}$$

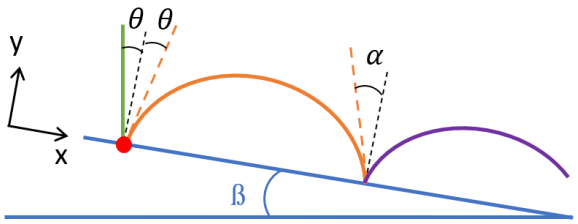


Figure 22. Splitting the movement of the bouncing ball

Subsequently, the contact with the ground (red point on figure 22) can be represented using a simple spring-damper model (see figure 23). It is assumed that the *Pymunk* [3] physics simulation used calculates similarly, as an elasticity factor can be defined in the settings and not all calculations of the physics engine could be retraced. To be able to describe the whole ball movement, this section mainly focuses on determining the impact velocity of the ball. This is where the weight force, the spring force and the damping force with their respective constants m (ball mass), c (spring stiffness) and d (damping constant) occur.

$$\begin{aligned} F_g + F_c + F_d &= m * \ddot{y} \\ m * \ddot{y} + d * \dot{y} - c * (r - y) &= -m * g \end{aligned} \tag{22}$$

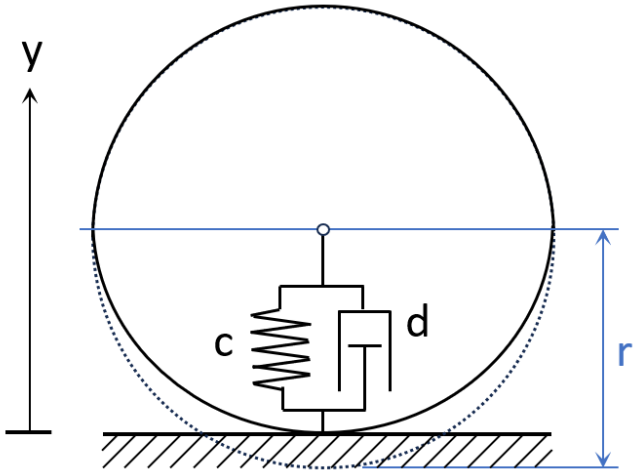


Figure 23. Spring-damper model to simulate the ball impact and bounce on the ground

After the impact, the next motion section (the orange part in figure 22) is assumed to be an oblique throw with the initial velocity v_0, which was calculated using equation 22 above. Additionally, the principle "angle of incidence = angle of reflection" is used to determine the angle of rebound θ from the ball; this corresponds to the ground slope β after the first ground contact.
The following equations of motion for X and Y are derived

from Newton's law:

$$m * \ddot{x} = m * g * \sin(\beta) \tag{23}$$

$$m * \ddot{y} = -m * g * \cos(\beta) \tag{24}$$

The equations can also be expressed as functions of time by integrating them. v_0 is the initial velocity of this third phase of motion, which was calculated above in the second part of the movement (impact on the ground).

$$x = \frac{g * \sin(\beta)}{2} * t^2 + v_0 * \sin(\theta) * t + x_0 \tag{25}$$

$$y = -\frac{g * \cos(\beta)}{2} * t^2 + v_0 * \cos(\theta) * t + y_0 \tag{26}$$

When the ball hits the ground for the second time after the "oblique throw", the orange motion phase ends and the purple phase begins. In between, there is another bounce, where equation 22 helps to determine the second rebound velocity.
The further movement is then described by the above equations 23 and 24. This requires the impact angle α, which can be calculated at the end of the orange phase as follows:

$$\tan(\alpha) = \frac{v_y}{v_x} = \frac{-g * t_{Auftreff} + v_{y,0}}{v_{x,0}} \tag{27}$$

$t_{Auftreff}$ can be calculated using the equation of motion in the y-direction by setting it to zero.

Finally the entire ball movement can be described step by step using the equations listed above. The rotation of the ball is determined as for the rolling case, by forming the angular momentum balance for each bounce, taking into account the friction force of the ground.
In the bouncing case, all degrees of freedom (movements in the X and Y directions, as well as the ball rotation) contain second-order terms. In addition, it becomes evident through the more complex equation 22 of the ball impact that the motion along the Y-axis is more complex here than along the ground.

D.2. Bouncing ball

D.2.1 Further evaluation and results

Another result for the bouncing ball is shown in figure 24. Here it can be seen that the ball position varies slightly depending on the algorithm. The biggest error is made by the diffusion network along the vertical: the ball is drawn about 9px too low.

In the table 11, the average results for all evaluation criteria and the three AI methods are shown.

In order to supplement the main section with a few more details on the respective AI networks, their strengths and weaknesses are now summarised. A few typical error patterns are also listed for each network.

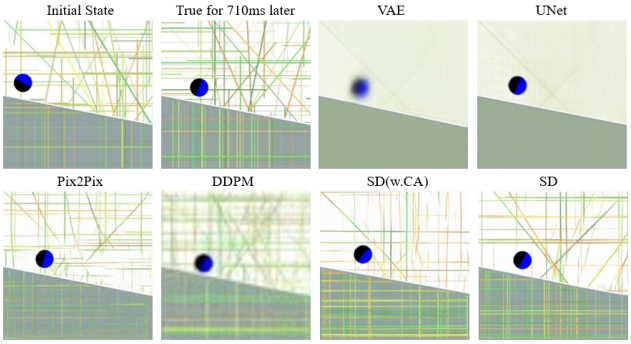


Figure 24. Predictions of the generative AI compared to the physics simulation for the rolling ball

Pix2Pix (GAN architecture):

- Most stable mapping of the bouncing ball problem: best results achieved relative to the number of errors
- Most error cases for the ball rotation: 15% of the generated images are not interpretable regarding the error metric. For the "valid" 85%, there is an average error of 17,2° with a standard deviation of 20,8°, reflecting the high variance in the results.
- Only one ball appears on the generated images in 93% of cases, while in the remaining 7%, no proper ball is represented. In these cases, either no ball is present on the image or the depicted shape is too fragmented and has no roundness at all.
- The network most frequently places the "target ball" in the X-direction (i.e. along the ground slope) behind the start ball. In fact, the generated image must always show the ball with a larger X-coordinate than on the input image, otherwise the correctness of the physics is not given. Only in 1% of the cases the target ball is in front of the start ball; otherwise all valid predictions correctly represent the rolling along the inclined surface.
- Good representation of the background: the line structures are most accurately represented for the horizontal and vertical lines. The diagonal segments are usually only partially drawn.
- Correlations between error and simulation parameters:
  - Position error in the X-direction increases the most with a larger time interval between input image and prediction, followed by an increasing ground inclination.
  - Position error in the Y-direction: a slight correlation is visible here between increasing start height of the ball and increasing error, although the ground inclination and the time interval have almost the same influence.
  - Ball angle deteriorates when the time interval between the start and target images increases.
  - No real correlation is observed for the roundness error.

UNet:

Table 11. Prediction results for the Pix2Pix, UNet, autoencoder and diffusion networks for the bouncing ball

Model	**Metric**	**Mean**	**Standard deviation**	**Error**	**Number of balls** 0 / >1 / Error	**Position to start ball** Ahead / Error
Pix2Pix	Position X	6.28	7.98	7%	7% / 0% / 0%	1% / 7%
	Position Y	11.7	12.8	7%		
	Rotation	17.2	20.8	15%		
	Roundness	0.56	0.14	10%		
	Ground slope	0	0	0%		
UNet	Position X	5.53	7.48	20%	19% / 0% / 1%	0% / 20%
	Position Y	10.8	12.2	20%		
	Rotation	15.2	22.9	30%		
	Roundness	0.74	0.21	35%		
	Ground slope	0	0	0%		
VAE	Position X	4.69	6.13	89%	20% / 0% / 68%	1% / 89%
	Position Y	6.25	6.94	89%		
	Rotation	31.0	39.8	97%		
	Roundness	0.90	0.14	99%		
	Ground slope	1.00	0.04	25%		
convAE	Position X	4.24	3.85	89%	11% / 0% / 86%	0% / 97%
	Position Y	6.08	5.93	97%		
	Rotation	12.2	8.61	99%		
	Roundness	1.06	0	99.9%		
	Ground slope	1.00	0.05	26%		
DDPM	Position X	7.91	9.04	3%	0% / 1% / 2%	6% / 3%
	Position Y	15.5	13.7	3%		
	Rotation	32.9	33.8	8%		
	Roundness	0.61	0.17	6%		
	Ground slope	0	0.08	3%		
SD(w.CA)	Position X	40.0	48.5	0%	0% / 0% / 0%	12% / 0%
	Position Y	24.8	22.9	0%		
	Rotation	61.1	52.5	21%		
	Roundness	0.53	0.15	2%		
	Ground slope	1.80	1.51	0%		
SD	Position X	8.55	11.9	0%	0% / 0% / 0%	5% / 0%
	Position Y	16.2	14.1	0%		
	Rotation	34.2	37.8	6%		
	Roundness	0.47	0.11	0%		
	Ground slope	0.14	1.08	0%		

- General: slightly better results in position and rotation compared to the Pix2Pix, if only the successfully analysable images are considered. Accuracy in the X-direction is about 1px better and rotation about 2° better.
- Algorithm stability: much more unstable: 20% to 35% of the predictions are not evaluable. Many images are generated that contain no ball or a shape that does not approximate a ball.
- **Target ball placement:** the network also places very well the target ball "behind" the start ball in terms of the X-coordinate. All valid result images comply with the physical correctness of rolling along the ground slope. Where this is not fulfilled, the images are not evaluable.
- Background depiction: blurry representation of the background. As seen in the result images, both the ground and the sky are depicted almost monotonously. The UNet seems unable to represent fine line structures; only very attenuated vertical lines can be guessed. Similarly, the

different colours of the lines are not shown: the ground appears uniformly gray and the sky light green.

- Correlations between error and simulation parameters:
  - Position error in the X-direction increases most with a larger time interval, followed by an increasing ground inclination (similar to the Pix2Pix)
  - Position error in Y-direction increases mainly with greater starting height of the ball, although the correlation remains relatively weak (as with the Pix2Pix)
  - Ball angle deteriorates when the time interval between the start and target image increases
  - For roundness, unlike the Pix2Pix, there is a dependence on the time interval; if this becomes larger, the error also increases.

Diffusion: Three different models were examined under the category of "diffusion": the DDPM, stable diffusion with cross-attention (SDw.CA) and without (SD). In general, the results of the DDPM and SD are very similar to each other, with the SD showing a little more stability; the SDw.CA performs significantly worse except for the mapping of ball roundness, which is very good.
For the DDPM network different approaches and settings were tested because the results of the first runs with 256px images were very poor. On one image, several balls appeared at randomly distributed positions or the prediction was noisy and contained artefacts (see figure 25). Since no

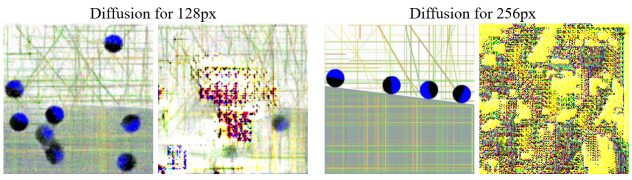


Figure 25. Typical predictions for the DDPM model trained with 128 and 256px images

approach or parameter set could be found to achieve better predictions with this algorithm, the image size was progressively reduced. This had a very positive influence on the training, making the result images from the 64px network comparable to the Pix2Pix and UNet approaches. Therefore, only results for 64px images are shown in this paper for the DDPM, which explains the blurriness caused by upscaling the predictions to 256px. With the stable diffusion approach there were no such resolution problems, so that the 256px images could be used directly.

- In general, the error criteria of the diffusion images are worse than those of the other analysed networks. However, depending on the error measure, between 4 and 7% fewer non-evaluable images are produced compared to the Pix2Pix and significantly fewer than with the UNet, which has the highest error rate.
- In the position, the accuracy is between 25-40% worse. Rotation proves to be the biggest weakness of the approach, as the error here is twice as high as with Pix2Pix and UNet. The roundness is mapped similarly to the other methods and even a bit better for the stable diffusion. This is the strength of this method, which delivers very high-quality images with round balls. The SD approach depicts the roundness the best without generating error images regarding this criterion.
- It is noticeable that the ground slope cannot be evaluated in approx. 3% of the images for the DDPM. Depending on the diffusion method, the mean error is also different from zero, which was not the case either with Pix2Pix or with UNet.
- Regarding the number of balls in the result images, the diffusion approach performs best. For the DDPM in 97% of the images, only one ball is visible (which is not even achieved by GAN at 93%) and the remaining 3% is divided between images with multiple balls and error images. With stable diffusion, there are even no images with several or no balls.
- The approach is a bit less reliable in terms of mapping the roll of the target ball along the ground slope. SDw.CA in particular performs worst with 12% of the cases where the target ball is located ahead the start ball along the X-axis, which is physically impossible. The DDPM and SD shows quite the same results with about 5 to 6% ahead target balls. Nevertheless, the predictions that correctly represent this relationship are not much worse with diffusion, as the error rate is reduced.
- The prediction images correctly represent the different colors as well as the line structures in the sky and ground. Due to the blurred image, the visual evaluation is worse for the DDPM, but all "image elements" are present. Both stable diffusion algorithms generate very high-quality images in which the colors, the background pattern and the ball itself are mapped very accurately.
- Correlations between errors and simulation parameters:
  - Generally, there are hardly any recognizable correlations with the simulation settings: only for the position error along the X axis and the ball rotation some relationships can be noticed.
  - Position error in X direction has a slight dependency on the time interval between input and target image (a larger interval leads to greater inaccuracy)
  - Ball angle deteriorates the further the ball of the input image has moved along the ground surface. For input images with a ball already in the middle of the image, the rotation is slightly worse. This correlation is slightly stronger for the SD approach compared to the DDPM.
  - No correlations are present for the Y position and ball roundness errors.

Auto encoder: Two approaches were examined for the autoencoders: the variational AE (VAE) and the convolutional AE (convAE). However, no evaluable results could be generated by the networks for the bouncing ball. Depending on the error criterion, between 89 and 99% of the predictions are evaluated as invalid (see results in Table 11). In general, it can be said that the obtained images are very blurred. In addition, the ball is rarely displayed as a single point but is often drawn out as if it were taking up several positions in the image. The ground surface is also not displayed cleanly as a single line; instead, it looks as if several slopes are depicted using different lines. There are also small white stripes in the ground at some images. These typical artefacts or errors of the autoencoder can be seen in Figure 26. Structurally, meaning in terms of colors and background pattern, the predictions look similar to the UNet results, as the networks are closely related in design.
Since so many generated predictions are considered invalid when evaluating the error criteria, the results for the bouncing ball in terms of position, angle and roundness are hardly relevant. Therefore, no correlations between simulation parameters and error measures are provided here.

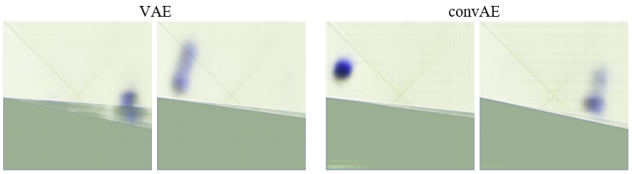


Figure 26. Typical error patterns with auto encoders (VAE or convAE) for the bouncing case - no clean representation of the ground surface and very blurred ball

In table 12 the detailed results of different training and evaluations are visible for the Pix2Pix and UNet approaches. Indeed for these two networks many runs were carried out to see how the models behave over multiple trainings. In the overview tables already shown in this paper, the mean values of all runs of one method are shown for each error criterion.

D.2.2 Typical error patterns

In this section, typical incorrect predictions of the analysed generative networks for the bouncing ball are presented to illustrate the above analysis and the error rate of the networks. Due to the poor results of the autoencoders, no error images are presented for this network structure (the typical appearance of the predictions has already been explained above). The visible errors or inaccuracies are described in the caption of the respective figures.

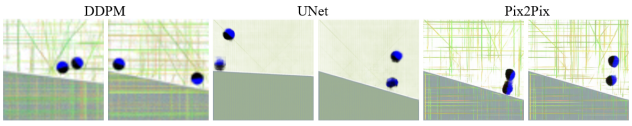


Figure 27. Several balls represented on the predictions of the DDPM, the UNet and the Pix2Pix algorithms.

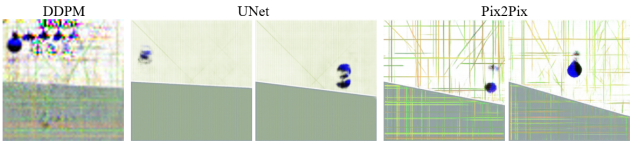


Figure 28. Typical artefacts that occur for different trained model

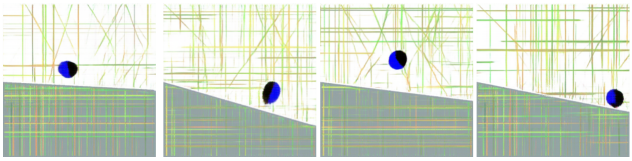


Figure 29. Not-round ball and imperfect color separation between both ball halves (only Pix2Pix predictions)

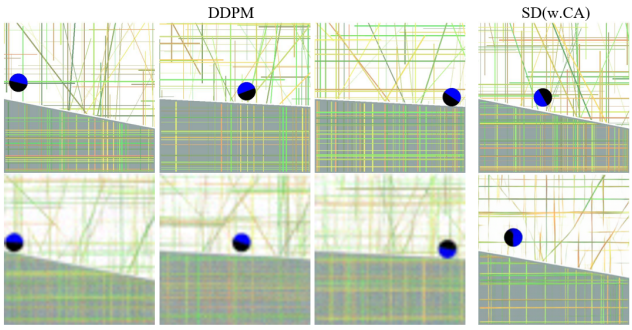


Figure 30. Position of the target ball ahead of the start ball along the X-axis (1st line: start ball; 2nd line: corresponding network prediction)

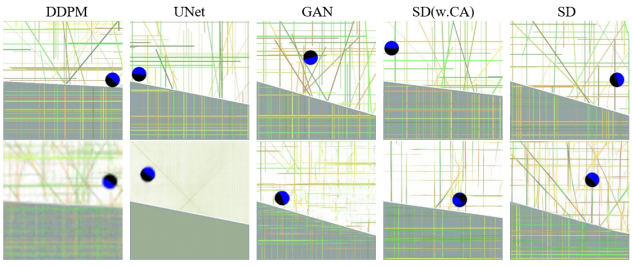


Figure 31. Errors concerning the ball position and rotation (1st line: true images; 2nd line: corresponding net prediction)

Table 12. Detailed results of each training run for the Pix2Pix (GAN) and UNet networks where multiple runs were carried out to get more representative mean values for the bouncing ball

	Rotation Mean / Std / Err	**Position X** Mean / Std	**Position Y** Mean / Std	**Position** Error	**Roundness** Mean / Std / Err
Pix2Pix	15.9 / 18.8 / 58 (4%)	6.46 / 8.27	12.2 / 12.9	6 (0%)	0.50 / 0.11 / 16 (1%)
	16.7 / 19.9 / 112 (7%)	6.31 / 8.17	12.1 / 13.0	14 (1%)	0.51 / 0.12 / 23 (1%)
	14.5 / 20.5 / 587 (37%)	5.71 / 7.25	10.3 / 11.9	433 (27%)	0.73 / 0.20 / 612 (38%)
	21.6 / 24.1 / 229 (14%)	6.65 / 8.23	12.4 / 13.3	14 (1%)	0.51 / 0.11 / 19 (1%)
UNet	15.1 / 23.4 / 495 (31%)	5.30 / 7.44	10.8 / 11.8	310 (19%)	0.73 / 0.22 / 542 (34%)
	15.9 / 23.7 / 398 (25%)	5.55 / 7.62	10.9 / 12.1	261 (16%)	0.75 / 0.21 / 540 (34%)
	15.2 / 24.1 / 458 (29%)	5.54 / 7.62	11.3 / 12.9	274 (17%)	0.74 / 0.22 / 520 (33%)
	14.5 / 20.5 / 587 (37%)	5.71 / 7.25	10.3 / 11.9	433 (27%)	0.73 / 0.20 / 612 (38%)

	Ground slope Mean / Std / Error	**Number of balls** 0 / >1 / Error	**Position to start ball** Ahead / Error
Pix2Pix	0 / 0 / 0 (0%)	3 (0%) / 1 (0%) / 0 (0%)	9 (1%) / 6 (0%)
	0 / 0 / 0 (0%)	12 (1%) / 2 (0%) / 0 (0%)	16 (1%) / 14 (1%)
	0 / 0 / 0 (0%)	416 (26%) / 0 (0%) / 17 (1%)	0 (0%) / 433 (27%)
	0 / 0 / 0 (0%)	14 (1%) / 0 (0%) / 0 (0%)	5 (0%) / 14 (1%)
UNet	0 / 0 / 0 (0%)	304 (19%) / 0 (0%) / 4 (0%)	3 (0%) / 310 (19%)
	0 / 0 / 0 (0%)	256 (16%) / 1 (0%) / 2 (0%)	3 (0%) / 261 (16%)
	0 / 0 / 0 (0%)	263 (16%) / 0 (0%) / 8 (1%)	3 (0%) / 274 (17%)
	0 / 0 / 0 (0%)	416 (26%) / 0 (0%) / 17 (1%)	96 (6%) / 44 (3%)

D.3. Rolling ball

D.3.1 Further evaluation and results

Result images of the predictions for the rolling ball are shown below on figure 32. For this simpler physics problem, there are only two variable parameters in the simulation: the ground slope and the position of the "start ball" on the image. Of course, predictions are also expected from the AI networks for any given time interval. Visually, the same advantages and disadvantages as with the bouncing ball can be recognized for the analysed AI algorithms (see section D.2.1). Here too, the DDPM network creates a 64px image, which is then scaled up to 256px. That is because the network trains better with the 64px images and delivers more "stable" results. The up-scaling explains the blurriness of the diffusion image.

In the table 13, the average results of several runs for the investigated AI networks for the rolling ball are shown.

The first noticeable point in the results is that the autoencoders provide evaluable predictions for the rolling case, even though they remain the most unstable method. In fact, the error images drop to around 3% for the position and regarding the ball angle and roundness a maximum of 34% invalid images are generated. Therefore, the results of this model structure can be discussed below. Otherwise it can be seen that the errors for the position are smaller compared to the bouncing case. This is especially evident for the Y-coordinate, where a significant reduction of the error

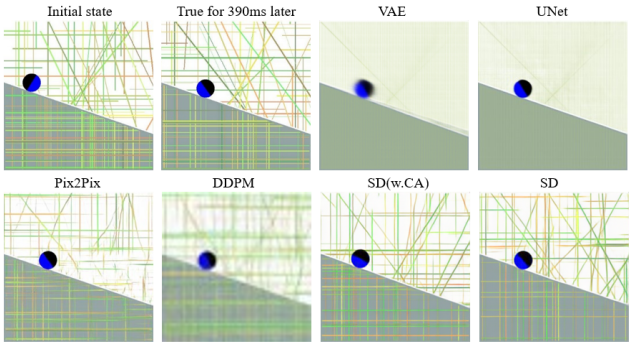


Figure 32. Predictions of the generative AI compared to the physics simulation for the rolling ball

can be seen, as the ball here "only" rolls along the ground surface and no longer moves in the Y-direction. The third physical error measure, which describes whether the ball is lying on the ground surface, is approximately the same for all AI methods, ranging between 97-99%. Only the autoencoders have difficulties to represent the ball rolling down on the ground. The ball roundness is mapped similarly to the bouncing ball, although it is much better with the UNet. Finally, the ball angle is represented about 2° worse by each algorithm, except for the stable diffusion models which are improving by approx. 2-4°. A few special features of the analysed AI approaches and differences compared to the bouncing ball are listed below.

Table 13. Prediction results for the Pix2Pix, UNet, autoencoder and diffusion networks for the rolling ball

Model	**Metric**	**Mean**	**Standard deviation**	**Error**	**Number of balls** 0 / >1 / Error	**Position to start ball** Ahead / Error	**Distance to ground** Error
Pix2Pix	Position X	4.26	8.36	1%	1% / 0% / 0%	0% / 1%	2%
	Position Y	1.61	3.31	1%			
	Rotation	20.9	35.2	13%			
	Roundness	0.56	0.13	3%			
	Ground slope	0	0	0%			
UNet	Position X	3.69	8.49	1%	3% / 0% / 0%	0% / 3%	3%
	Position Y	1.4	3.63	1%			
	Rotation	16.1	32.7	11%			
	Roundness	0.53	0.15	4%			
	Ground slope	0	0	0%			
VAE	Position X	5.54	16.2	3%	3% / 0% / 0%	1% / 3%	25%
	Position Y	2.44	6.26	3%			
	Rotation	21.5	36.0	34%			
	Roundness	0.70	0.20	25%			
	Ground slope	1.10	0.33	0%			
convAE	Position X	4.57	9.47	2%	2% / 0% / 0%	0% / 2%	12%
	Position Y	2.10	3.80	2%			
	Rotation	21.4	38.2	20%			
	Roundness	0.65	0.21	12%			
	Ground slope	1.11	0.32	0%			
DDPM	Position X	7.38	8.56	1%	0% / 0% / 1%	3% / 1%	3%
	Position Y	2.34	3.39	1%			
	Rotation	34.2	38.6	10%			
	Roundness	0.74	0.16	6%			
	Ground slope	0	0.05	1%			
SD(w.CA)	Position X	24.1	33.2	1%	1% / 0% / 0%	3% / 1%	2%
	Position Y	7.65	11.6	1%			
	Rotation	59.1	52.1	12%			
	Roundness	0.55	0.13	2%			
	Ground slope	1.06	0.39	0%			
SD	Position X	7.36	14.7	0%	0% / 0% / 0%	1% / 0%	1%
	Position Y	2.57	5.32	0%			
	Rotation	29.9	40.4	4%			
	Roundness	0.53	0.11	1%			
	Ground slope	0.02	0.13	0%			

Pix2Pix (GAN architecture):

- Significant reduction of "error images" created by the network that cannot be analysed regarding all error criteria except for the ball rotation. Here, the ball angle is still not recognisable for 13% of the images.
- Slight increase in the rotation error but strong increase in the standard deviation compared to the bouncing case. This is the error measure that is most poorly mapped by the Pix2Pix model.
- With 99% reliability, exactly one ball is represented on the predictions of the AI network (6% improvement over the bouncing case). Similarly, in 99% of cases, the target ball is placed behind the start ball in the X-direction (7% improvement).
- Correlations between error and simulation parameters:
  - Position errors in both directions (X and Y) increase the most with increasing ground slope, followed by a larger time interval. Here, the X and Y errors show the same dependencies on the physics simulation settings.
  - Ball angle deteriorates with greater ground inclination and when the time interval between the start and target image increases.
  - For the roundness error, unlike the bouncing case, there is a slight dependence on the starting position of the ball: the further the ball of the input image has moved along the ground surface, the less round the ball is represented.

UNet:

- The same insights and improvements apply to the UNet as to the Pix2Pix model concerning the position and rotation of the ball, the number of balls and the placement in X-direction of the target ball behind the input ball.
  With this approach, the error images have decreased the

most compared to the bouncing case: the largest reduction of non-analysable images from 35% to 4% is observed for the ball roundness.
- The roundness, which is worse with the bouncing ball, achieves the best error measure for the rolling case.
- Overall, the UNet provides the best results for the rolling ball, slightly ahead of the Pix2Pix algorithm. However, the background of the predicted images (ground and sky) remains blurred, so that the GAN performs better in terms of overall appearance.
- Correlations between error and simulation parameters:
  - Position errors in both directions increase as the time interval increases; there is a significant dependence here. There is also a smaller correlation with the increasing ground slope, but it is weaker.
  - For ball rotation, the time interval is more dominant compared to the GAN, followed by the ground slope. As the parameters increase, the inaccuracy in the prediction of the ball angle also increases.
  - For the roundness error, the same dependence as for the bouncing case is observed: a larger time interval between input and target images results in a poorer ball representation.

Diffusion: Even in the rolling case, the best results for the DDPM model were obtained with 64px images, explaining the blurriness of the predictions due to up-scaling.

- Like the bouncing ball, the error criteria of the diffusion method are worse than for GAN and UNet. Here, stable diffusion with cross-attention delivers the worst results, with the network performing two or three times worse than the other diffusion methods, depending on the criterion. The DDPM and SD approaches differ only slightly in the results, with the SD model demonstrating greater stability with a very low error rate and a better representation of the ball roundness. Compared to the bouncing case, the position error in the Y-direction is mainly reduced, as with the other algorithms, while the other error metrics remain nearly identical.
- Both the ball position (in both directions) and the angle are represented with about twice the inaccuracy compared to Pix2Pix and UNet for the DDPM and the SD model. The roundness is about 35% worse.
- Regarding the mapping of the ground slope, the same is observed as for the bouncing case: very few error images and a small standard deviation around the mean value of 0 are present for the DDPM, while the UNet and the GAN architecture draw the ground almost "perfectly". The stable diffusion approaches even show a mean error greater than zero.
- With a 99% reliability going to almost 100% for the SD, exactly one ball is shown on the predictions of the diffusion networks and the few remaining predictions are not evaluable. Here, diffusion performs best, as about 1% of the Pix2Pix and 3% of the UNet predictions do not contain a ball.
- Regarding the correct rolling of the ball on the ground surface, 3% of predictions represent the target ball ahead of the input ball along the X-direction, which is physically impossible. Here, both other methods discussed above perform slightly better with almost 0% target balls ahead. However, the diffusion network, like the Pix2Pix and the UNet, correctly depicts the ball on the ground surface in 97 to 99% of cases.

  Overall the SD model shows the best results concerning the three criteria expected to verify the physical correctness.
- Correlation between error and simulation parameters:
  - Position error in the X direction most increases with a larger time interval. There is also a smaller correlation with the increasing ground slope.
  - Position error in the Y direction increases the most with increasing ground slope, followed by a larger time interval.

    Concerning the position errors along both axes the DDPM algorithm shows the strongest dependencies.
  - Ball angle deteriorates the most when the time interval between the start and finish images increases. There is also a slightly weaker correlation with the ground slope.
  - No correlations are present for the roundness error.

Autoencoder: For the rolling ball, the two examined autoencoder approaches generate evaluable results. With a maximum of 34% invalid images for ball angle predictions and an error rate of 3% for the position, the quality of the predictions is significantly improved. Additionally, the average error metrics are close to those of the Pix2Pix model and even slightly better than the diffusion networks. Nonetheless, autoencoders remain the most unstable structure for modeling the ball problem, as they generate the highest number of non-evaluable images. The following observations can be made regarding the VAE and convAE approaches:

- The convAE network generally provides slightly better results than the VAE. The error rates of convAE are a bit lower, as well as reduced average errors and standard deviations. However, the mapping of the ground and the representation of the ball angle are nearly identical between the two methods.
- As with the other models, the ball's position is represented less accurately along the X-axis compared to the Y-direction for the autoencoders.
- The ball rotation is learned as effectively as with the Pix2Pix approach, but the error rate increases by 50% to more than double in the case of the VAE algorithm.

- A notable observation is that the ball's roundness is represented worse compared to all other approaches examined, with significantly more invalid images. Similarly, the ground slope is learned less accurately by both models. With the SDw.CA method, the autoencoders are the only models showing a mean error greater than zero.
- Regarding physical correctness, these models generate the highest number of images without any ball present (3%). Additionally, in 12% (convAE) and 25% (VAE) of the cases, the ball is not depicted on the ground surface, although this should always be the case for a rolling ball on an inclined surface. This error criterion stands out most significantly compared to the other models, where only 1 to 3% of the images show the ball not in contact with the ground.
- Correlation between error and simulation parameters:
  - Position errors in both directions increase the most as the time interval increases. A slight dependency can be observed to the ground slope, which generates more inaccuracy when getting greater (more pronounced for the Y-direction).
  - Concerning the ball rotation the correctness of the model is worser when the time interval between start and target images increases. There is also a slightly weaker correlation with the ground slope.
  - For the roundness error, a correlation to the position of the ball on the image can be observed. The further the ball of the input image has moved along the ground surface, the less round it is depicted. This is the only model that shows a dependency between the roundness error and this simulation parameter.

In table 14 the detailed results of different training and evaluations are visible for each of the three analysed AI approaches. In the overview tables already shown in this paper, the mean values of all runs of one method are shown for each error criterion.

D.3.2 Typical error patterns

In this section, typical incorrect predictions of the analysed networks for the rolling ball are presented to illustrate the above analysis and the error rate of the networks. The visible errors or inaccuracies are described in the caption of the respective figures.

Finally, the evaluation of the rolling case can be concluded with a brief look at the physics equations. As already indicated in the main part of this paper, the largest errors can be traced back to the parts of the equation that contain a double derivative. The prediction of the x-coordinate and the ball angle are the most error-prone and also occur in the equations with x and ψ double derivatives. In the Y direction, the significant improvement in the position prediction can be explained by the fact that there is no movement along this axis and therefore the complex calculation parts of the bouncing case are omitted.

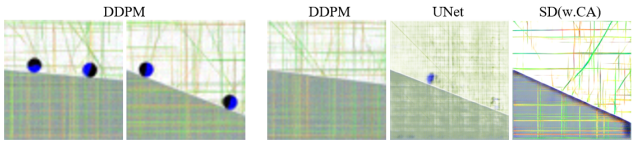


Figure 33. Several or no balls are present in the predictions (left: several balls only occur with the diffusion network; right: no ball present)

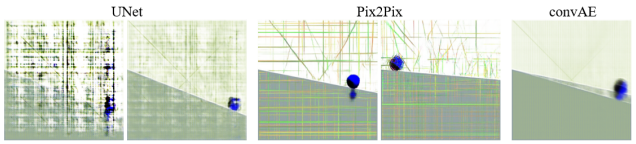


Figure 34. Typical artefacts that occur for different trained model

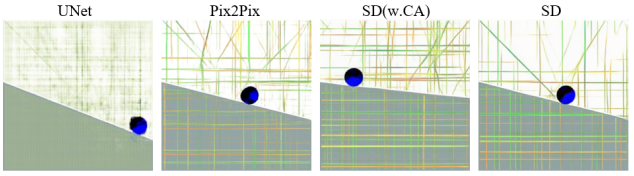


Figure 35. Imperfect color separation between both ball halves (occurs mostly for the Pix2Pix model)

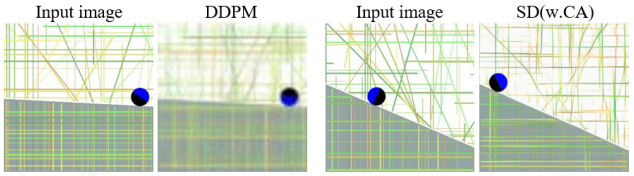


Figure 36. Position of the target ball ahead of the start ball along the X-axis (mostly for diffusion). The predicted ball is slightly to the left of the starting ball position on the generated samples.

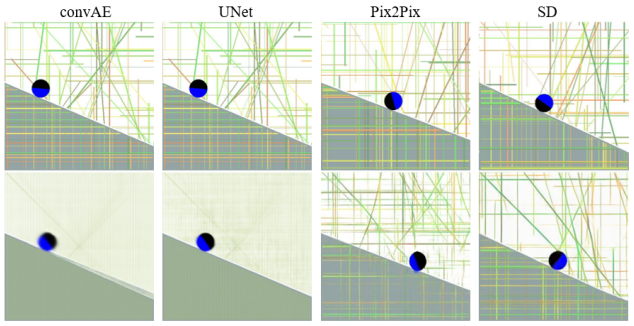


Figure 37. Errors concerning the ball position and rotation (1st line: true images; 2nd line: corresponding network prediction)

Table 14. Detailed results of each training run for the Pix2Pix and UNet networks where multiple runs were carried out to get more representative mean values for the rolling ball

	Rotation Mean / Std / Err	**Position X** Mean / Std	**Position Y** Mean / Std	**Position** Error	**Roundness** Mean / Std / Err
Pix2Pix	21.7 / 36.0 / 199 (11%)	4.17 / 8.28	1.58 / 3.24	16 (1%)	0.60 / 0.14 / 34 (2%)
	24.4 / 36.7 / 385 (21%)	4.35 / 8.14	1.67 / 3.32	26 (1%)	0.62 / 0.15 / 85 (5%)
	21.8 / 35.4 / 377 (21%)	4.92 / 9.36	1.91 / 3.82	21 (1%)	0.57 / 0.14 / 63 (4%)
	16.2 / 32.9 / 121 (7%)	3.80 / 8.43	1.38 / 3.26	3 (0%)	0.50 / 0.11 / 16 (1%)
	20.2 / 34.8 / 98 (5%)	4.07 / 7.60	1.49 / 2.92	14 (1%)	0.51 / 0.11 / 16 (1%)
UNet	16.8 / 33.5 / 176 (10%)	3.69 / 8.33	1.30 / 3.18	39 (2%)	0.52 / 0.15 / 62 (3%)
	15.7 / 32.1 / 194 (11%)	3.59 / 8.35	1.33 / 3.26	40 (2%)	0.51 / 0.14 / 66 (4%)
	16.2 / 33.6 / 187 (10%)	3.77 / 8.69	1.40 / 3.46	23 (1%)	0.53 / 0.15 / 62 (3%)
	15.6 / 31.5 / 203 (11%)	3.71 / 8.59	1.57 / 4.61	9 (1%)	0.55 / 0.15 / 68 (4%)

	Ground slope Mean / Std / Err	**Number of balls** 0 / >1 / Error	**Position to start ball** Ahead / Error	**Distance to ground** Error
Pix2Pix	0 / 0 / 0 (0%)	16 (1%) / 0 (0%) / 0 (0%)	5 (0%) / 16 (1%)	19 (1%)
	0 / 0 / 0 (0%)	13 (1%) / 0 (0%) / 0 (0%)	8 (0%) / 26 (1%)	34 (2%)
	0 / 0 / 0 (0%)	21 (1%) / 0 (0%) / 0 (0%)	0 (0%) / 21 (1%)	35 (2%)
	0 / 0 / 0 (0%)	36 (2%) / 0 (0%) / 2 (0%)	0 (0%) / 38 (2%)	38 (2%)
	0 / 0 / 0 (0%)	14 (1%) / 0 (0%) / 0 (0%)	2 (0%) / 14 (1%)	14 (1%)
UNet	0 / 0 / 11 (1%)	37 (2%) / 1 (0%) / 1 (0%)	0 (0%) / 39 (2%)	40 (2%)
	0 / 0 / 0 (0%)	36 (2%) / 0 (0%) / 4 (0%)	0 (0%) / 40 (2%)	40 (2%)
	0 / 0 / 1 (0%)	44 (2%) / 0 (0%) / 8 (0%)	0 (0%) / 52 (3%)	52 (3%)
	0 / 0 / 0 (0%)	57 (3%) / 0 (0%) / 7 (0%)	0 (0%) / 64 (4%)	64 (4%)

E. Datacard

PhysicsGen	This assembled dataset comprises three distinct physical-based image-to-image translation tasks, totaling 300,000 samples across diverse domains. The datasets include: **Urban Sound Propagation:** Expanding on research into urban sound propagation, this dataset contains 25,000 data points extracted from 10 different cities. Each city is represented by 2,500 locations within a 500m x 500m area, utilizing OpenStreetMap imagery where buildings are marked with black pixels and open spaces with white pixels. **Lens Distortion Correction:** Derived from the CelebA dataset, this dataset focuses on correcting lens distortion in facial images. **Dynamics of rolling and bouncing movements:** This dataset captures the physical dynamics of rolling and bouncing balls, intended for studies in motion prediction. Each dataset facilitates the development of generative models by providing high-resolution (256x256) imagery and diverse scenarios.
DATASET LINK	DATA CARD AUTHOR(S)
https://doi.org/10.5281/zenodo.10609793	**Anonymous for Blind Review**

Authorship

Publishers

PUBLISHING ORGANIZATION(S)	INDUSTRY TYPE(S)	CONTACT DETAIL(S)
Anonymous for Blind Review	Anonymous for Blind Review	**Anonymous for Blind Review**

Funding Sources

INSTITUTION(S)	FUNDING OR GRANT SUMMARY(IES)
Anonymous for Blind Review	Anonymous for Blind Review

Sound Propagation - Dataset Overview

DATA SUBJECT(S)

Data about natural phenomena

Data about places and objects

DATASET SNAPSHOT

Size of Dataset	~5 GB
Number of Instances	~100000
Training	19908 x 4
Evaluation	3732 x 4
Test	1244 x 4

Additional Notes: The dataset is segmented into four distinct subsets, each tailored to explore specific aspects of sound propagation in urban environments: Baseline, Reflection, Diffraction, and Combined.

CONTENT DESCRIPTION

A data point in this dataset consists of two main components: an urban layout image from OpenStreetMap and a corresponding sound distribution map. The urban layout image is a 500m x 500m area depiction where buildings are marked in black and open spaces in white. The sound distribution map generated using NoiseModelling v4.0, illustrates the sound dynamics within that urban environment at resolutions of 512x512 or 256x256.

The dataset comprises four subsets: **Baseline** for basic sound behavior, **Reflection** examining sound wave interactions with surfaces, **Diffraction** focusing on sound navigating around objects, and **Combined** which merges reflection, diffraction and changing environmental factors like temperature and humidity.

Dataset Version and Maintenance

MAINTENANCE STATUS

Regularly Updated

(New versions of the dataset have been or will continue to be made available.)

VERSION DETAILS

Current Version: 2.0

Last Updated: 06/2024

Release Date: 06/2024

MAINTENANCE PLAN

Feedback:
Anonymous for Blind Review

Sound Propagation - Example of Data Points

PRIMARY DATA MODALITY	SAMPLING OF DATA POINTS	DATA FIELDS

Multimodal

- Image Data
- Geospatial Data
- Tabular Data

Field Name	Field Value	Description
lat	float	Latitude of the sound measurement location.
long	float	Longitude of the sound measurement location.
db	Object	Key-value pairs of sound levels in decibels for a given frequency (lwd{fqz}).
soundmap	string	Path to 256x256 resolution sound distribution image.
soundmap_512	string	Path to 512x512 resolution sound distribution image.
osm	string	Path to Open Street Map image showing urban layout.
temperature	float	Temperature (°C) at the location.
humidity	float	Humidity (%) at the location.
yaw	float	Orientation of the noise source. Can be empty.
sample_id	int	Unique identifier for the data point.

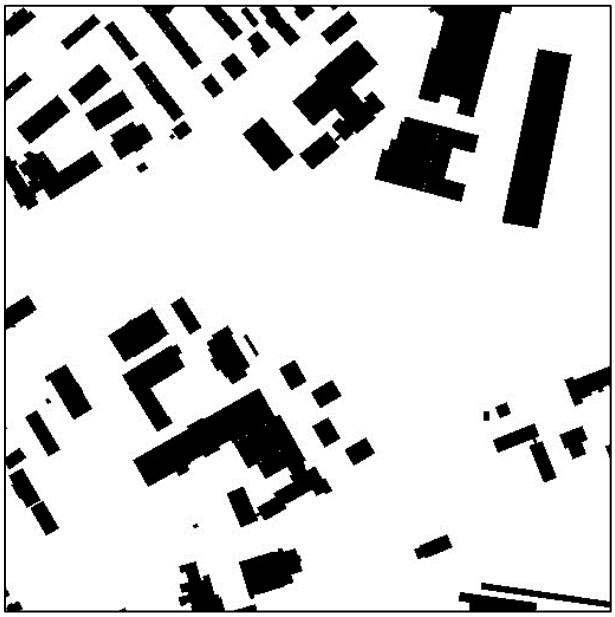
TYPICAL DATA POINT


```
``` {
  "lat": 48.030229082138526,
  "long": 11.367773397906852,
  "db": {"lwd500": 69},
  "soundmap":
  "./soundmaps/256/0_LEQ_256.png",
  "soundmap_512":
  "./soundmaps/512/0_LEQ_512.png",
  "osm":
"./buildings/osm_23747.png",
  "temperature": 12,
  "humidity": 35,
  "yaw": None,
  "sample_id": "23747"
  } ```
```

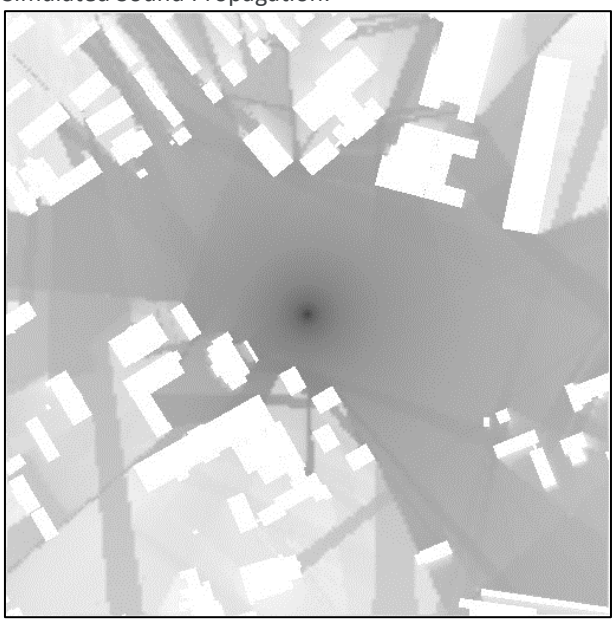

EXAMPLE OF DATA POINT

Below is an example of an OSM and Simulated Sound Propagation pair:

OSM:



Simulated Sound Propagation:

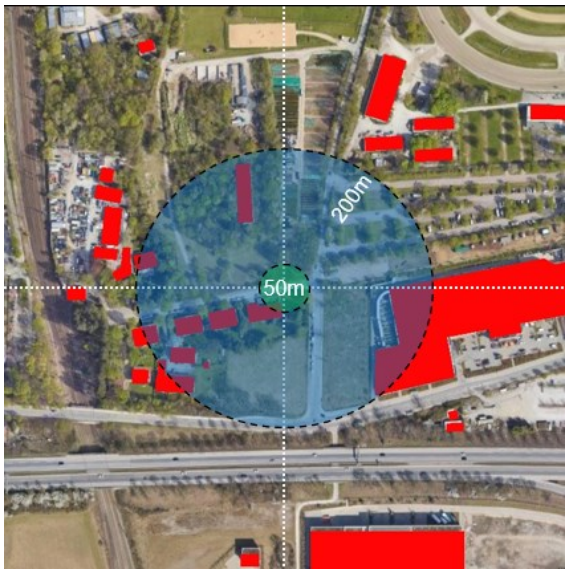

```

Sound Propagation - Provenance

Collection

METHOD(S) USED	METHODOLOGY DETAIL(S)	SOURCE DESCRIPTION(S)
- API - Physical Simulation Framework	**Overpass API** **Source:** The Overpass API is a read-only API that serves up custom selected parts of the OSM map data. It acts as a database over the web: the client sends a query to the API and gets back the data set that corresponds to the query. **Platform:** https://overpass-api.de/ **Is this source considered sensitive or high-risk?** [Yes / **No**] **Dates of Collection:** [10 2023 - 12 2024] **Primary modality of collected data:** Geospatial Data **NoiseModelling v4.0** **Source:** An advanced simulation tool engineered for accurate modeling of sound dynamics within urban environments. **Platform:** https://github.com/Universite-Gustave-Eiffel/NoiseModelling **Is this source considered sensitive or high-risk?** [Yes / **No**] **Dates of Collection:** [10 2023 - 12 2024] **Primary modality of collected data:** Geospatial Data	**OSM Buildings:** This source provides images from Open Street Map (OSM) that depict urban layouts, specifically focusing on buildings within cities. In these images, black pixels represent buildings, and white pixels indicate open spaces. **Sound Propagation:** This component of the dataset involves simulated sound distribution images around urban centers, where the noise source is placed at the center.

Collection Criteria

DATA SELECTION	DATA INCLUSION	DATA EXCLUSION
Location Sampling: The locations are randomly sampled across 10 cities/areas: ["Hamburg", "Hannover", "Augsburg", "Bonn", "Muenchen", "Schwerin", "Berlin", "Paris", "Stuttgart", "Aachen"]	**Enough Obstacles:** At least 10 Buildings within a circle r=200m around the sound source. **No Obstacle to close:** No Building within a r=50m circle around the sound source. **Additional Notes:** 	**Additional Notes:** If the Data Inclusion criteria is not met, the data is excluded. No additional exclusion criteria are introduced.

Lens Distortion - Dataset Overview

DATA SUBJECT(S)

Data about natural phenomena

DATASET SNAPSHOT

Size of Dataset	~2 MB
Number of Instances	100000
Training	40000 x 2
Evaluation	7500 x 2
Test	2500 x 2

Additional Notes: The dataset is segmented into two subsets for p1 and p2.

CONTENT DESCRIPTION

This dataset provides the lens parameters corresponding to images from the CelebA dataset in a structured CSV format. Intended for use in computer vision research focusing on the effects of different lens configurations on image characteristics, the dataset organizes parameters across training, evaluation, and testing subsets. It includes scripts available on our GitHub repository for researchers to reproduce the image samples within the bounds of copyright law.

Dataset Version and Maintenance

MAINTENANCE STATUS

Regularly Updated

(New versions of the dataset have been or will continue to be made available.)

VERSION DETAILS

Current Version: 1.0

Last Updated: 06/2024

Release Date: 06/2024

MAINTENANCE PLAN

Feedback:
Anonymous for Blind Review

Lens Distortion - Example of Data Points

PRIMARY DATA MODALITY	SAMPLING OF DATA POINTS	DATA FIELDS
Tabular Data		

Field Name	Field Value	Description
label_path	string	Path to the label file associated with the image sample.
fx	float	The focal length of the camera lens used to capture the image.
k1, k2, k3	float	Coefficients representing the radial distortion introduced by the lens.
p1, p2	float	Coefficients representing the tangential distortion of the lens.
cx	float	The x-coordinate of the principal point of the image, which is the point on the image sensor where the lens is focused.
distortion_path	string	Path to the file containing distortion image.

TYPICAL DATA POINT


```
``` {
  "label_path":
"labels_50k/label_0.jpg",
  "fx": 200,
  "k1": 0.0,
  "k2": 0.0,
  "k3": 0.0,
  "p1": 0.13393540720902974,
  "p2": 0.0,
  "cx": 128,
  "distortion_path": "true/y_0.jpg"
  }```
```
```

Lens Distortion - Provenance

Collection

METHOD(S) USED	METHODOLOGY DETAIL(S)	SOURCE DESCRIPTION(S)
- Python Package	**OpenCV** **Source:** OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. The library provides a common infrastructure for computer vision applications and accelerates the use of machine perception in commercial products. It contains more than 2500 optimized algorithms, including a comprehensive set of both classic and state-of-the-art computer vision and machine learning techniques. **Platform:** https://opencv.org/ **Is this source considered sensitive or high-risk?** [Yes / **No**] **Dates of Collection:** Not applicable (continuously updated) **Primary modality of collected data:** Image Processing Data	**OSM Buildings:** This source provides images from Open Street Map (OSM) that depict urban layouts, specifically focusing on buildings within cities. In these images, black pixels represent buildings, and white pixels indicate open spaces.

Dynamics of rolling and bouncing movements - Dataset Overview

DATA SUBJECT(S)

Data about natural phenomena

DATASET SNAPSHOT

Size of Dataset	~7 GB
Number of Instances	~75,000
Rolling - Training	27840
Rolling - Evaluation	50
Rolling - Test	1800
Bouncing - Training	44835
Bouncing - Evaluation	58
Bouncing - Test	1600

Additional Notes: The dataset is segmented into two distinct subsets, for a rolling ball and a bouncing ball.

CONTENT DESCRIPTION

The third physics problem investigated in this publication is the movement of a rolling or bouncing ball. The aim here is to evaluate the ability of generative AI to map kinematic movements of physics: to investigate how well the networks can predict the position and rotation of the ball along an inclined surface for a defined time interval after the input image.

Dataset Version and Maintenance

MAINTENANCE STATUS

Regularly Updated

(New versions of the dataset have been or will continue to be made available.)

VERSION DETAILS

Current Version: 1.0

Last Updated: 06/2024

Release Date: 06/2024

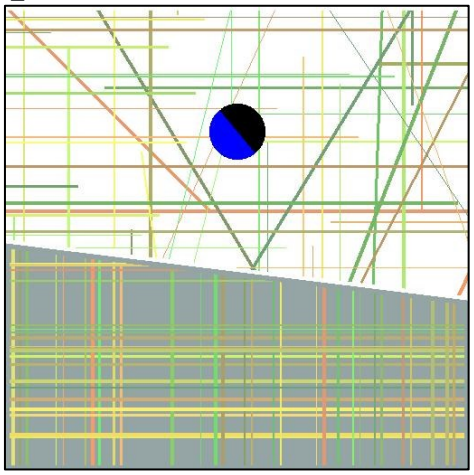
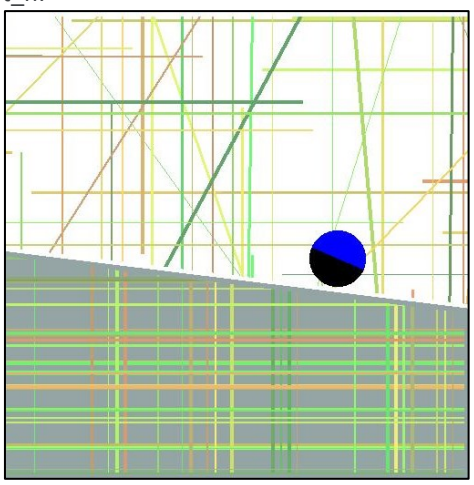
MAINTENANCE PLAN

Feedback:
Anonymous for Blind Review

Dynamics of rolling and bouncing movements - Example of Data Points

PRIMARY DATA MODALITY	SAMPLING OF DATA POINTS	DATA FIELDS
Multimodal - Image Data - Tabular Data		

Field Name	Field Value	Description
ImgName	float	The filename of the image pair capturing the rolling ball at a specific instance.
StartHeight	int	The initial height from which the ball starts.
GroundIncli	int	The angle of incline of the ground on which the ball is rolling, expressed in degrees.
InputTime	int	The simulation time when the ball begins its movement.
TargetTime	int	The timestamp at which the prediction should be made, indicating when the model estimates the ball will reach a predetermined target point or condition.

TYPICAL DATA POINT	EXAMPLE OF DATA POINT
`` { "ImgName": "17", "StartHeight": 0, "GroundIncli": -3, "InputTime": 99, "TargetTime": 5910 } ```	Below is an example of an OSM and Simulated Sound Propagation pair: t_0:  t_n: 

Dynamics of rolling and bouncing movements - Provenance

Collection

METHOD(S) USED	METHODOLOGY DETAIL(S)	SOURCE DESCRIPTION(S)
- Physical Simulation Framework	**Pymunk** **Source:** Pymunk is a physics simulation library based on Chipmunk, which provides a fast, lightweight 2D rigid body physics library for Python. It enables the creation and manipulation of physics simulations in a simple manner, offering precise control over elements like mass, gravity, and friction. **Platform:** http://www.pymunk.org/ **Is this source considered sensitive or high-risk?** [Yes /**No**] **Primary modality of collected data:** Physical Simulation Data	

Motivations & Intentions

Motivations

PURPOSE(S)	DOMAIN(S) OF APPLICATION	MOTIVATING FACTOR(S)
Research	*`Generative Models`, `1-step Physic Simulation`, `Sound Propagation`, `Machine Learning`*	The release of this dataset is driven by the potential of generative learning models to understand and simulate complex physical phenomena. Our dataset of 300,000 image-pairs supports research into three specific physical simulation tasks: urban sound propagation, lens distortion correction, and the dynamics of rolling and bouncing balls. By providing these data, along with baseline evaluations, we aim to address several fundamental research questions: - Learning Capability: Can generative models accurately learn and replicate complex physical relationships from input-output image pairs? - Efficiency Gains: What computational speedups can be achieved by substituting traditional, differential equation-based simulations with generative model predictions?

Intended Use

DATASET USE(S)	RESEARCH AND PROBLEM SPACE(S)
Safe for research use	The dataset targets three key physical phenomena: urban noise propagation, lens distortion, and the dynamics of rolling and bouncing balls. It is crafted to develop models capable of: - **Urban Noise Propagation**: Predicting sound distribution around urban buildings. - **Lens Distortion**: Correcting optical distortions in photographs. - **Dynamics of rolling and bouncing movements:** Simulating motion trajectories under various conditions.