ManiWAV: Learning Robot Manipulation from In-the-Wild Audio-Visual Data (Appendix)

Zeyi Liu[1] Cheng Chi[1,2] Eric Cousineau[3] Naveen Kuppuswamy[3]
Benjamin Burchfiel[3] Shuran Song[1,2]

[1]*Stanford University* [2]*Columbia University* [3]*Toyota Research Institute*

https://maniwav.github.io/

1 Method Details

1.1 Audio Latency Calibration

Similar to using a clapperboard in film production, we tap on the contact microphone and match the time between the frame when the finger is observed to be in contact with the microphone and the corresponding audio signal captured by the contact microphone. A visual illustration is shown in Fig. 1. The audio is received about 0.06s after the image is received. As a result, the total audio latency is 0.17 + 0.06 = 0.23s, where 0.17s is the calibrated image latency following the approach in [1].

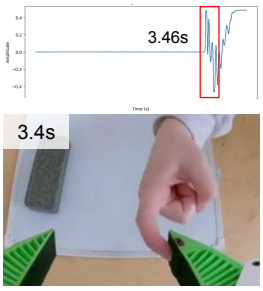


Fig 1

1.2 Model Details

Image Augmentation. Each image is randomly cropped with a 95% ratio and then resized to its original resolution in the replay buffer. To make the learned model more robust to different lighting conditions at test time, we apply ColorJitter augmentation with brightness 0.3, contrast 0.4, saturation 0.5, and hue 0.08.

Spectrogram Parameters. There are several parameters that control the time and frequency resolution of the resulting spectrogram image when converting the audio waveform to a spectrogram. The window length is the length of the fixed intervals in which STFT divides the signal; it controls the time and frequency resolution of the spectrogram. Hop length is the length of the non-intersecting portion of the window lengths. The smaller the hop length is, the more times a particular audio segment will present in STFT, and the more elongated the time-axis of the resulting spectrogram will be. For a sample rate of 16kHz, we use the default window length, 400, as recommended by torchaudio. It is recommended that the hop length is around ½ of the window length; we choose 160 to slightly augment the signal pattern along the time-axis since contact signals are usually sparser in time compared to nature sounds or speech.

Transformer Encoder. We use one transformer encoder layer with 8 heads to fuse the vision and audio features. We set the feedforward dimension to 2048 and the dropout ratio to 0.0.

End-to-End Training Details. For each task, the entire model is end-to-end trained on 2 NVIDIA GeForce RTX 3090 GPUs for 60 epochs, with a batch size of 64. We use the AdamW optimizer with lr=1e-4, betas=[0.95, 0.999], eps=1.0e-8, weight_decay=1.0e-6, and apply EMA (Exponential Moving Average) on the weights.

2 Evaluation Details

2.1 In-the-Wild Data Collection

We collect 274 in the wild data for the bagel flipping task in total, including 52 in a conference room, 37 and 44 in two kitchens, 46 in an office, 76 on lounge tables, and 19 in a cafe, using 4 different pans. Examples of the environments are shown in Fig. 2.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

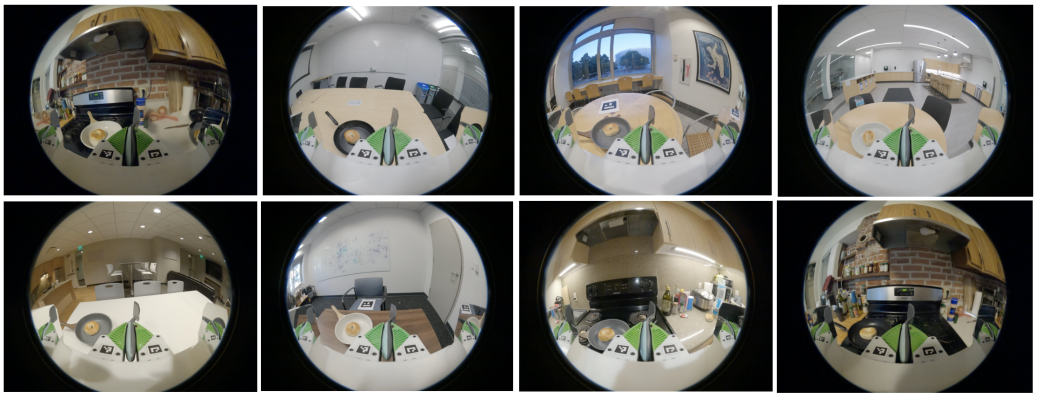


Fig 2: Example Scenes in the In-the-Wild Dataset.

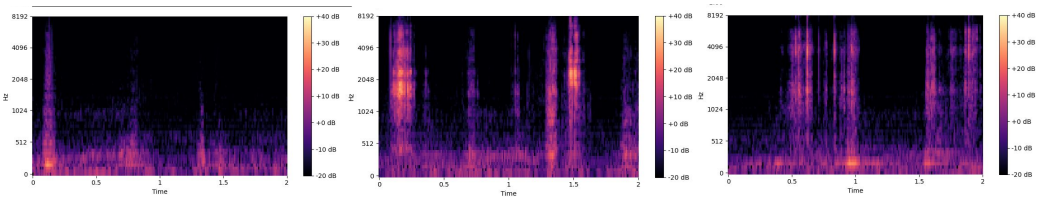


Fig 3: We visualize the audio spectrogram in the pouring task when the robot 'shakes' the cup to sense whether there are objects inside the cup. (a) shows the audio feedback of an empty cup, (b) and (c) shows the audio feedback for a cup with dice and with screws inside, respectively.

2.2 Result Details

2.2.1 Pouring Task

A breakdown of the success rate for each substep in the pouring task is shown in Tab. 1. 'Grasp' is successful if the robot grasps the white cup stably in its end effector, 'Pour' is successful if the robot pours all objects in the white cup to the pink cup on the table, 'Place' is successful if the robot places the white cup on the table after pouring. By using audio feedback to infer the object state (whether there are objects in the cup or not), our method is able to reliably guide the policy to pour objects and place the cup, whereas the baselines either never execute the pour action or the place action, resulting in low substep success rate. We also find that the policy can generalize to unseen objects such as screws. In Fig. 3, we visualize the audio spectrogram when the robot 'shakes' the cup. We can observe that the spectrogram for an empty cup is distinctive from a non-empty cup, and a cup with screws generates a similar audio pattern as a cup with dice upon shaking. We hypothesize that the audio features for the cup with screws and the cup with dice are also close in the audio feature space, leading to similar policy behavior. Videos of the policy rollouts can be found on the project website.

	Grasp	Pour	Place
OURS	90	**100**	**90**
Vision only	100	0	8.3
1s audio	91.7	30	0
10s audio	100	60	16.7

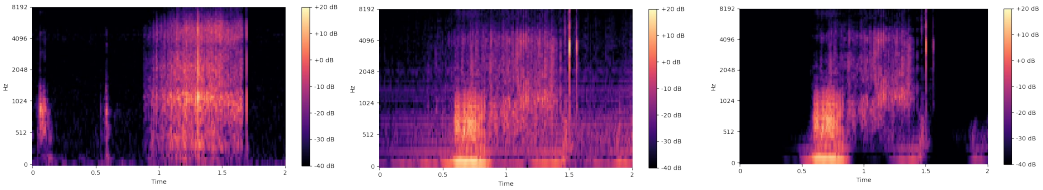
Table 1: Success Rate Breakdown.

2.2.2 Taping Task

A breakdown of the success rate for each substep in the taping task is shown in Tab. 2. 'Touch' is successful if the robot slides along the tape while maintaining contact, 'Sense' is successful if the robot chooses the correct tape, 'Pick' is successful if the robot successfully grasps the tape. 'Place' is successful if the

	Touch	Sense	Pick	Place
OURS	90	**90**	80	80
Vision only	80	30	90	80
Env Mic	80	30	90	40
Noise Reduction	80	40	90	90

Table 2: Success Rate Breakdown.



(a) Training signal (b) Real time signal (c) Signal after noise reduction

Fig 4: Spectrogram Visualization.

robot successfully places the tape on top of the wires. By leveraging audio feedback to infer the object surface material (whether the tape is a 'hook' or 'loop'), our method is able to reliably guide the policy to choose the correct tape, whereas the baselines make random decisions, as shown in the 'Sense' step success rate. Videos of the policy rollouts can be found on the project website.

Noise Reduction Algorithm. We use the non-stationary noise reduction method introduced in [2]. The algorithm computes a spectrogram from the audio waveform, and then applies an IIR filter forward and backward on each frequency channel to obtain a time-smoothed version of the spectrogram. A mask is then computed based on the spectrogram by estimating a noise threshold for each frequency band of the signal/noise. Finally, a smoothed, inverted version of the mask is applied to the original spectrogram to cancel noise. In our experiments, we apply this algorithm directly to the real-time audio signals before feeding it to the model for inference. In Fig. 4, we show spectrogram visualization of the real-time signal (b) and the signal after reduction (c). Even though the noise reduction seems to successfully remove most of the robot noises and some other background noises, it does not preserve the original signal well and still results in a domain gap as compared to the training signal (a).

2.2.3 Wiping Task

We listed out each test scenario in the wiping task evaluation and success (1) / failure (0) for each method.

	Test Scenario	OURS	Vision only	MLP fusion	Noise masking	No noise aug
0	T1 (square)	1	1	0 (incomplete)	0 (press)	1
1	T1 (heart)	1	1	0 (incomplete)	1	0 (press)
2	T1 (circle)	1	0 (incomplete)	1	1	1
3	T3 (star)	1	1	1	0 (press)	0 (press)
4	T3 (+1 inch)	1	0 (float)	0 (incomplete)	1	0 (incomplete)
5	T3 (+5 inch)	1	0 (press)	0 (incomplete)	0 (press)	0 (press)
6	T3 (changing height)	1	0 (float)	0 (float)	0 (press)	1
7	T3 (changing height)	1	0 (press)	0 (incomplete)	0 (press)	0 (press)
8	T2 (white noise)	1	1	0 (press)	1	1
9	T2 (white noise)	1	1	1	1	1
10	T2 (construction noise)	1	1	1	0 (press)	0 (press)
11	T2 (music)	0 (float)	1	0 (incomplete)	1	0 (incomplete)
12	T1 (board orientation)	0 (float)	0 (float)	0 (float)	0 (float)	0 (incomplete)
13	T1 (board position)	1	1	0 (incomplete)	1	1
14	T3 (unseen eraser)	1	0 (float)	0 (press)	0 (press)	0 (press)
15	T3 (unseen eraser)	1	0 (incomplete)	1	1	0 (press)
16	T3 (-1 inch)	1	0 (float)	1	1	0 (float)
17	T3 (-1 inch)	1	0 (float)	0 (incomplete)	0 (press)	0 (press)
18	T3 (-2 inch)	0 (float)	0 (float)	1	1	0 (float)
19	T3 (-2 inch)	1	0 (float)	0 (incomplete)	0 (press)	1
	Success rate	85%	40%	35%	50%	35%

Table 3: Wiping Test Scenario Breakdown.

'Float' means the robot keeps floating above the board without contacting the board before it wipes off the shape. 'Press' means the robot exerts too much force downward and causes the gripper to bend against the board. 'Incomplete' means the robot fails to follow the shape, either stops wiping early or wipes in the wrong location.

As we can see from Tab. 3, 'Float' and 'Press' are most common in the [Vision only] baseline, especially when the table height is different than training, likely due to the fact that it's insufficient to infer contact from the top-down view wrist-mount camera image alone (as shown in Fig. 5).

The most common failure case in [MLP fusion] is 'incomplete', where the robot stops wiping and releases the eraser before the shape is completely wiped off. We hypothesize that this is because simply fusing vision and audio features with MLP layers loses information that's crucial for inferring the stage of the task.

Without noise augmentation, the policy exhibits various unexpected behaviors including 'press', 'incomplete', and 'float', because of the big domain gap between training and testing. The policy achieves better performance by simply masking out the robot noise frequency range in the [Noise masking] baseline, however, it still fails half of the time and most of the failure cases are 'pressing' too hard against the board.

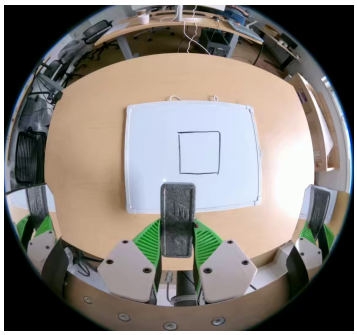


Fig 5: Camera view in the wiping task.

2.2.4 Flipping Task

We listed out each test scenario in the bagel flipping task evaluation and success (1) / failure (0) for each method.

	Test Scenario	OURS	Vision only	MLP policy	ResNet	AVID
0	T1	1	0 (lose)	1	0 (poke)	1
1	T1	1	0 (lose)	0 (displace)	1	0 (lose)
2	T1	1	0 (poke)	1	0 (displace)	0 (displace)
3	T1	1	0 (lose)	0 (poke)	1	1
4	T1	1	1	1	1	0 (lose)
5	T1	0 (lose)	0 (lose)	1	0 (displace)	1
6	T1	1	0 (lose)	1	1	1
7	T1	1	0 (poke)	0 (displace)	0 (poke)	1
8	T1	1	1	1	0 (displace)	0 (poke)
9	T1	1	1	0 (lose)	1	1
10	T1	1	1	1	0 (displace)	1
11	T1	1	0 (poke)	0 (poke)	0 (displace)	1
12	T1	1	0 (poke)	1	1	1
13	T1	1	0 (lose)	0 (lose)	0 (poke)	1
14	T2 (clap)	1	0 (poke)	1	1	0 (displace)
15	T2 (construction noise)	1	0 (poke)	1	0 (stuck)	1
16	T3 (unseen height)	1	1	1	1	0 (displace)
17	T3 (unseen height)	0 (lose)	0 (poke)	0 (lose)	0 (displace)	0 (lose)
18	T3 (unseen height)	1	0 (lose)	1	0 (poke)	0 (displace)
19	T3 (unseen height)	1	0 (displace)	0 (lose)	0 (displace)	0 (displace)
	Success rate	90%	25%	60%	35%	55%

Table 4: Flipping Test Scenario Breakdown.

'Poke' means that the robot pokes the spatula on the side of the bagel instead of inserting the spatula between the pan and the bottom of the bagel. 'Lose' means the robot loses contact with the bagel before it is flipped.

As a result, the bagel falls back to its original side. 'Displace' means that the spatula is displaced in the robot end effector as compared to its initial pose, as a result of the robot keeps moving down instead of switching to slide the spatula along the bottom of the pan.

For all T1 scenarios, we randomize the robot's initial pose and object positions (e.g., bagel and pan). We can observe that the [Vision only] policy does not generalize well across initial configurations and most failure cases are either poking on the side of the bagel (since it's hard to infer from the image alone if the spatula is contacting the bottom of the pan), or losing contact with the bagel early before it can be flipped.

Using a [ResNet] and [AVID] audio encoder results in the spatula to 'displace' most of the times, likely because the model is not sensitive enough to the sound feedback of spatula touching the bottom of the pan, and as a result the robot keeps moving downward and causes the spatula to displace.

References

[1] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[2] T. Sainburg. timsainb/noisereduce: v1.0, June 2019. URL https://doi.org/10.5281/zenodo.3243139.