A. Additional Details of AutoGaze Model Design

Model architecture. AutoGaze contains a convolutional vision encoder, a visual connector, and a transformer decoder. The convolutional encoder contains one 2D convolutional layer with spatial kernel size of 16 to embed each patch, and one 3D convolutional layer with spatial and temporal kernel sizes of 3 to extract the spatiotemporal vision features of each frame based on the current frame and previous two frames. Note that the vision encoder is causal. The visual connector bridges between the vision encoder output and the transformer decoder input, adding positional embeddings to the output visual features from the vision encoder and passing them to the decoder. The positional embeddings are added in each frame separately, such that each token in each frame is aware of its spatial position in the frame. The transformer decoder uses the same architecture design as LLaMA 3 [23] but with only four layers. The decoder takes in each frame's visual tokens and decodes the patch indices. It also predicts the reconstruction loss of the current frame at each step using a linear decoding head. The vocabulary of the decoder only contains all the possible patch indices in a frame. We use four scales, i.e., 32×32, 64×64, 112×112, and 224×224. Since the patch size is 16, the number of all possible patches (i.e., the vocabulary size of the decoder) is 4 + 16 + 49 + 196 = 265. The hidden dimension of the whole model is 192.

Instantiation of the reconstruction objective. AutoGaze is trained to select as few patches as possible while keeping a certain level of reconstruction loss (Eq. 2). The reconstruction objective includes a distance function $L(\cdot, \cdot)$ and a video reconstruction model $\texttt{Recon}(\cdot)$. We obtain the reconstruction model by taking an MAE [36] pre-trained on images and fine-tuning it on videos with the same masked auto-encoding objective such that it can reconstruct a video from partially observed patches. The resulting model is similar to VideoMAE [78] except that the self-attention layers are block-causal, i.e., the model reconstructs each frame based on only the current and previous frames. This is important because the gazing model is also causal and we should not train it to optimize a reconstruction loss that depends on the future. This also allows us to calculate the reconstruction loss of each frame based on previous gazed patches up until any step at that frame, which we use to supervise the reconstruction loss prediction at each step. For the distance function, we use a combination of pixel reconstruction loss and perceptual loss [43, 107], i.e., a weighted sum of ℓ_1 loss in the pixel space and ℓ_2 loss on the frame-wise DINOv2 [55] and SigLIP2 [79] embeddings between the reconstructed and the original video. The weights for ℓ_1 loss, DINOv2 embedding loss, and SigLIP2 embedding loss are 1, 0.3, and 0.3 respectively.

B. Additional Details of AutoGaze Training Pipeline

NTP pre-training. We pre-train AutoGaze on about 250K videos with paired gazing sequences. We train for 150 epochs, with a batch size of 256 and a learning rate of 5e-4.

RL post-training. We use GRPO algorithm that is a simplified and on-policy version of the original version, i.e., we remove the KL regularization term and use the same policy at each step as the old policy in the original algorithm. We use a group size of 12. When calculating the advantage for each gazing step, we count in the negative reconstruction loss for the current frame as well as the subsequent frames. The reconstruction loss for each frame is discounted based on the number of gazing steps between the current gazing and the last gazing of that frame. The discounting factor we use is 0.995. We also follow Dr. GRPO to remove the std normalization and the sequence length normalizer in GRPO. During RL training, we anneal the temperature when rolling out the gazing prediction from 1 to 0.01. At each step of training, instead of running VideoMAE on all the frame to get the reconstruction reward for every frame, we instead just randomly sample 2 frames to reconstruct and only use the reconstruction rewards for those two frames for RL training in order to improve training efficiency. When rolling out gazing prediction during training, we set a reconstruction loss threshold of 0.7, let AutoGaze to predict one gazing sequence based on the threshold, record the number of gazing for each frame, and then roll out a group of gazing sequences with the same number of gazing for each frame for GRPO loss. This ensures every gazing sequence in the same GRPO group has the same number of gazing for each frame, such that the reconstruction rewards between sequences can be fairly compared. We train for 3 epochs, with a batch size of 256 and a learning rate of 5e-4.

Training data. For the training data, we first collect videos from datasets including Ego4D [33], 100DoH [67], and InternVid [88], covering both exocentric and egocentric natural videos. We also create artificial videos that simulates camera motions by placing a window on an large image, slide the window to random directions, and take the content within the sliding window as a video. We create videos like this from high-resolution image datasets including SA-1B [44] and IDL [8], covering both natural and text-rich videos. In total, we end up with collecting ∼800K videos. For pre-training, we collect gazing sequences for a subset of ∼250K videos. Specifically, for each video, we first randomly sample the gazing ratios for each frame, and then search the best gazing sequence that optimizes the reconstruction loss under the sampled gazing ratio, using the greedy search algorithm described in Sec. 3.2. When sampling the gazing ratio, we first randomly sample the average gazing ratio of the whole video from an exponential distribution between 0.02 and 0.2.

Then, given the average gazing ratio of all the frames, we sample the gazing ratio for each frame using a Dirichlet distribution with alpha of 10 for the first frame and 3 for the rest of the frames, such that the gazing ratio is biased towards the first frame. This is to simulate the real distribution of gazing where the first frame is usually gazed at more since it contains completely new information without any history context.

C. Additional Details of HLVid Benchmark

We collect 268 QA on videos of autonomous driving and household scenarios scraped from YouTube. Each video has 4K resolution and a large portion of them have 5 minutes of duration. We design the QAs such that every question needs high-resolution perception at at least 1K - 2K resolution to solve. We also review the QAs such that questions from the same video are asking about different details, and the answer to each question is not ambiguous, i.e., there is only one correct answer and there is no other correct answers appearing in other frames in the video. As as comparison, existing benchmarks mostly only focus of long videos but not high resolution. For example, LongVideoBench [93] and EgoSchema [54] contain videos with an average duration of 473s and 180s separately, but their questions mostly only need low resolution to solve. Our benchmark, instead, requires both long and high-resolution video understanding to solve. See Figure 13 for examples of HLVid's videos and questions.

D. Additional Details of Analyses

Section 4.1 explores what AutoGaze pays attention to in videos. Below, we provide further implementation details for the optical flow and patch detail analyses.

Optical flow. To analyze whether AutoGaze selects moving patches more often than static patches, we use the image pairs and optical flow map in the Flying Chairs dataset [22]. Since the data only contains forward optical flow, it only assigns object motion to the pixels of the first frame, not the second frame. Therefore, we compute backward flow by finding each source pixel's resulting location and assigning the inverse optical flow there as well (occluded/invalid values are set to zero). The final flow map that we use is a pixel-wise maximum over the forward and backward optical flows. Given this map, we crop all possible patches at each gazing scale and compute the maximum optical flow in that patch.

Patch detail. We compute patch detail for Section 4.1 using the variance of the Laplacian for each image. Specifically, we compute the patch "detailedness" by convolving each video frame with a 3x3 Laplace filter

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

and computing the variance of the values in each possible patch. Lower variance corresponds to smoother/more constant coloring, while high variance corresponds to busier textures (e.g., stripes, text).

E. Additional Quantitative Results

How large of a reconstruction loss is tolerable for downstream video understanding? In Sec. 4.2, we benchmark the efficiency of ViTs and MLLMs with AutoGaze. However, since the efficiency depends on how many patches to select for each video, which further depends on the reconstruction threshold, we first need to pinpoint the appropriate reconstruction loss threshold, i.e., the reconstruction loss threshold that leads to little or no downstream performance drop. To do this, we set different reconstruction loss thresholds and compare the MLLM performance. Tab. 5 shows results averaged over four benchmarks [30, 47, 93, 94]. Compared to the baseline without gazing, reconstruction loss below 0.7 leads to performance drop consistently less than 0.5% across videos with different numbers of frames. This is supported by reconstruction visualization under different losses (Fig. 11), where a loss > 0.7 usually results in visible artifacts. Balancing performance and efficiency, we choose 0.7 as the threshold for MLLM experiments.

Table 5. **What reconstruction loss is tolerable for MLLM performance?** We find a loss of 0.7 has little performance drop.

Recon. Loss	64 Frames	128 Frames	256 Frames
No Gaze	59.1	60.1	60.5
0.6	59.0	60.0	60.7
0.7	58.6	59.7	60.3
0.8	57.8	58.4	59.0
1.0	56.3	56.7	57.2

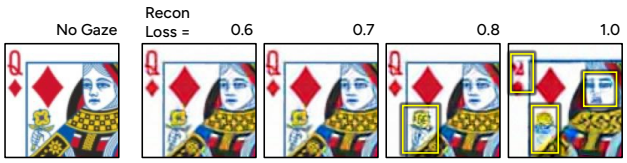

Figure 11. **Reconstruction quality under varying loss thresholds.** Outlined in `recon_loss`$= 0.8, 1.0$ are noticeable visual defects.

Efficiency gain of ViTs and MLLMs on streaming videos. In Sec. 4.2, we benchmark the ViT and MLLM efficiency with AutoGaze. However, the benchmarking is conducted on static videos, i.e., the models can see the full video beforehand such that it can split each video into multiple clips and process in a batchified way. However, lots of real-world applications calls for streaming video understanding, i.e., the ViT and MLLM needs to process frames one by one. To

this end, we also benchmark the ViT and MLLM efficiency under this situation, where they process frames one by one and we measure the maximum FPS they can process frames at. Results are shown in Fig. 12. We test on videos with different FPS and resolution, and compare the maximum FPS of ViT/MLLM with or without AutoGaze. We can see that ViTs and MLLMs with AutoGaze usually achieve an FPS that is up to $\sim$16$\times$ higher. This enables real-time processing for ViTs and MLLMs which is infeasible without AutoGaze (e.g., real-time ViT encoding of 10FPS videos at resolution higher than 500, and real-time MLLM processing of 3 FPS videos at 1K resolution).

F. Additional Qualitative Results

Figs 14 - 27 showcase examples of AutoGaze applied to various video domains and OOD use cases, including lectures, sports live stream, cartoons, film clips, picture-in-picture videos, fisheye lens security footage, warehouse surveillance camera, nighttime driving, black-and-white movies, robot arm demonstrations, and split-view videos. Finally, we provide another example of AutoGaze continuing to track moving objects even when an object is swapped in the middle of the video (Fig. 28). Refer to each figure caption for particular capabilities demonstrated in that example.

G. Limitations

We identify two main limitations with AutoGaze. First, it does not account for most camera motion; as a scene pans in some direction, it will still subsample patches but not necessarily ignore patches that are redundant up to a shift. See Fig. 29 for an example of this limitation. Second, our model cannot anticipate future frames according to knowledge of physics; though our VideoMAE is causal, it is not trained to have "intuitive physics" knowledge (e.g., knowledge that a falling ball will keep falling in the next frame). We illustrate this limitation in Fig. 30 by providing several full video frames of a ball in free fall, and visualizing the VideoMAE's reconstruction of subsequent frames.

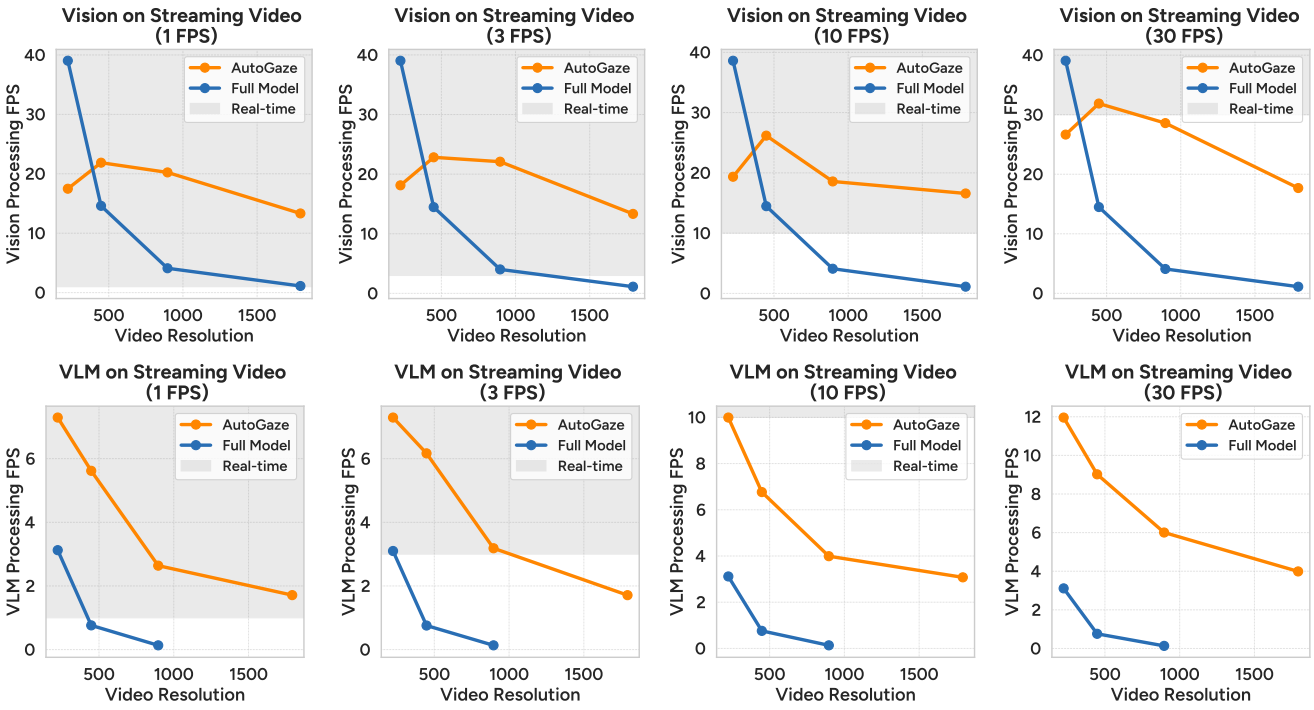

Figure 12. **Efficiency on streaming videos.** We compare the ViT and MLLM efficiency on streaming videos with or without AutoGaze. We measure the maximum FPS of processing different types of videos (i.e., different FPS and resolution). For each plot, the gray area denotes the region where real-time processing of the video is achieved.

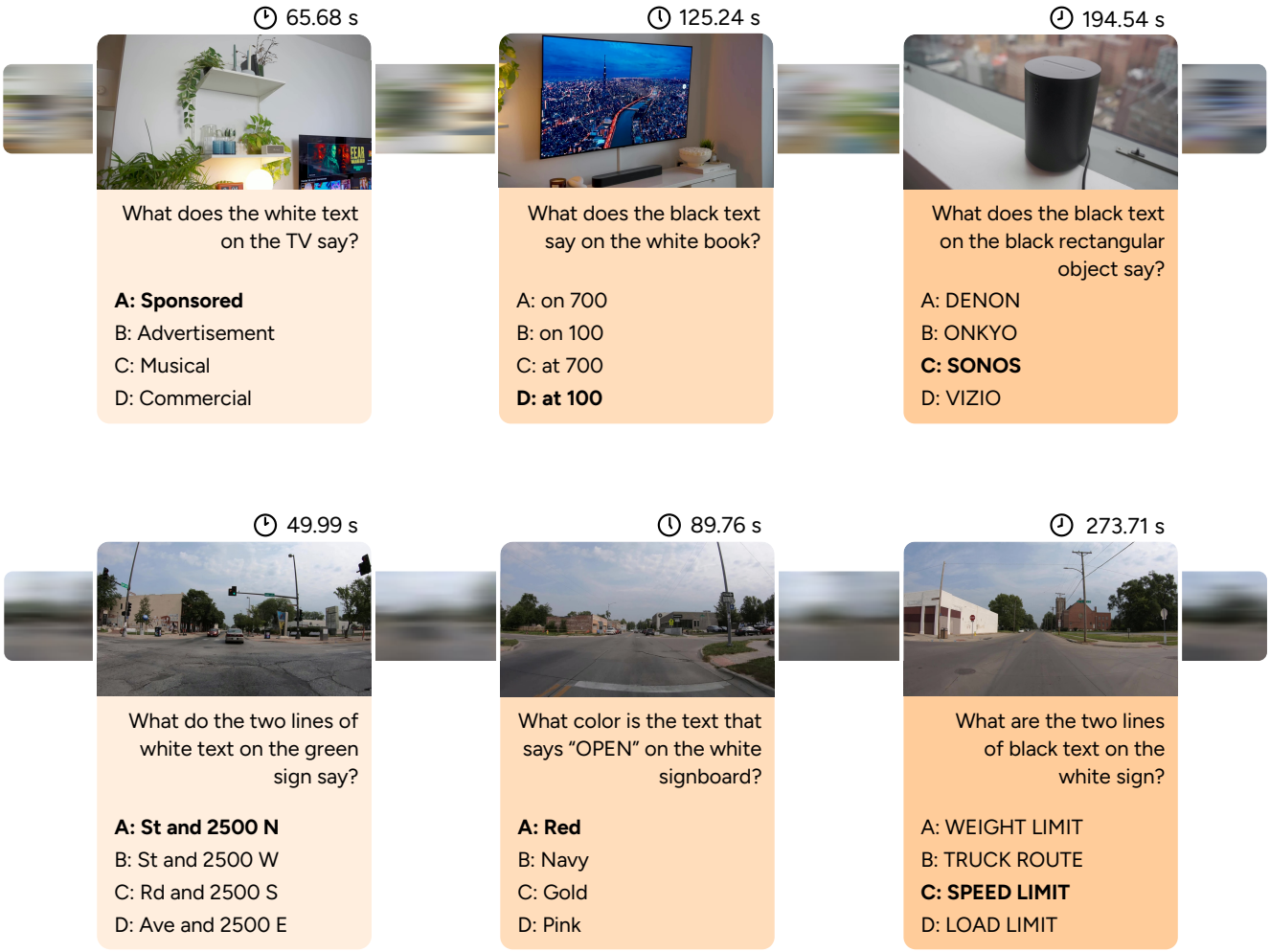

Figure 13. **Examples of HLVid benchmark.** Each sample from HLVid consists of a 5-minute long video at 4K resolution, along with multiple-choice questions that require high-resolution video encoding to answer. The answers to HLVid's questions can be found at any spatiotemporal location, emphasizing long-context video understanding. Our video content is diverse and includes house tours (top), city driving (bottom), nature videos, etc.

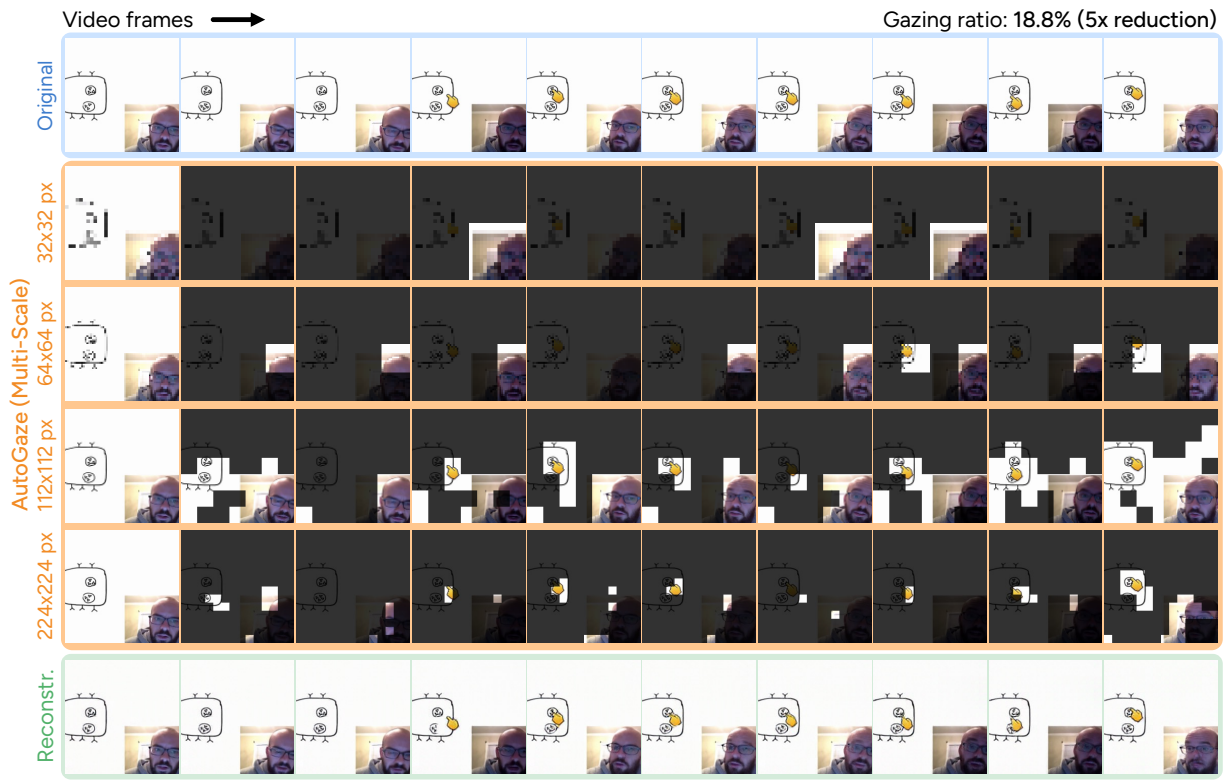

Figure 14. **Picture-in-picture whiteboard lecture.** After the first frame, AutoGaze focuses on the moving cursor and the lecturer's face.

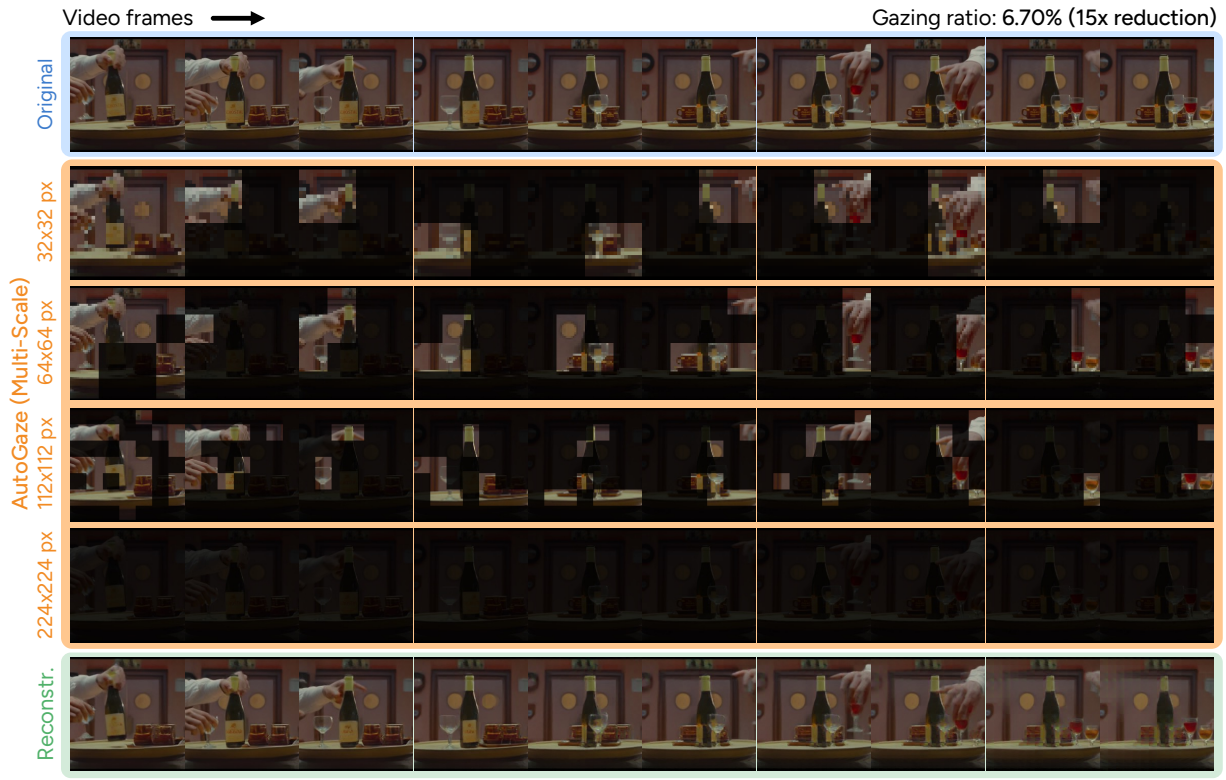

Figure 15. **Film clip.** AutoGaze selects minimal patches to track the drink bottles and glasses as they are moved by hands a rotating plate.

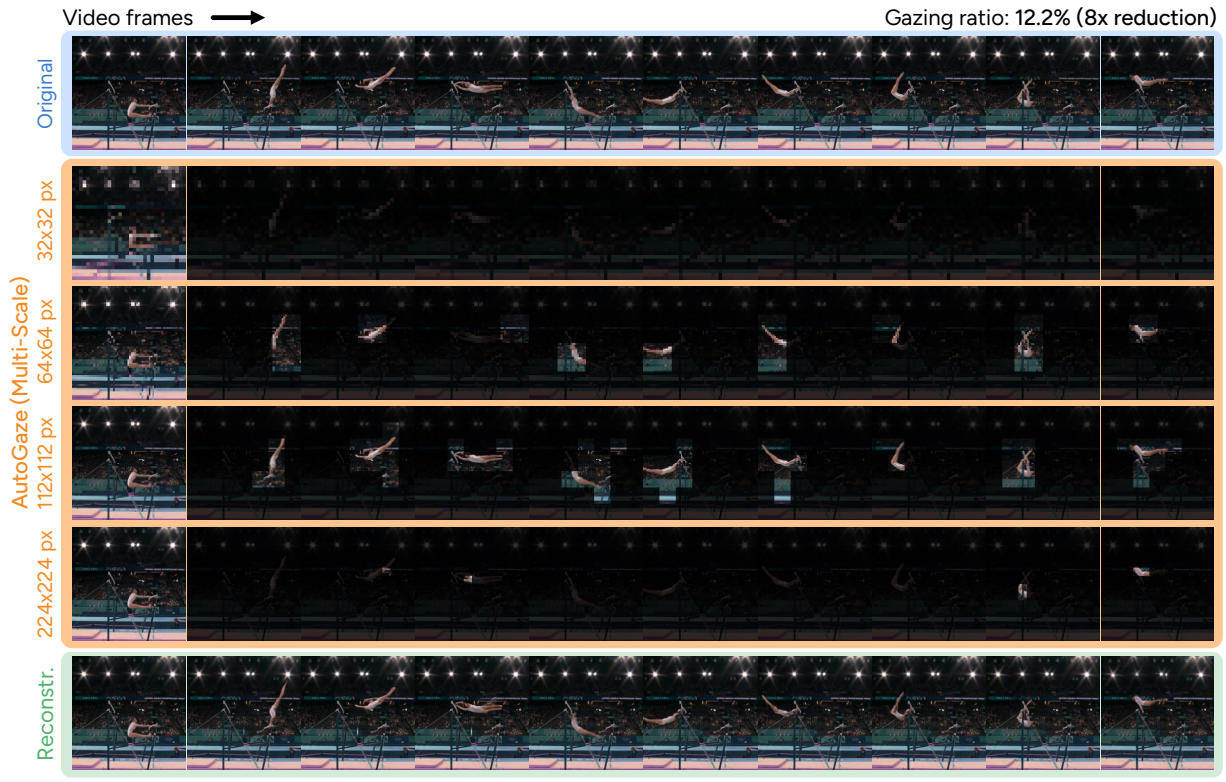

Figure 16. **Gymnastics clip.** AutoGaze tracks the gymnast across the uneven bars, using finer scales when appropriate.

Figure 17. **Claymation cartoon.** AutoGaze captures small movements (blinking), scene changes, and enough patches to reconstruct text.

Figure 18. **Claymation cartoon.** AutoGaze skips the finest scale as it is not needed to achieve the specified reconstruction loss.

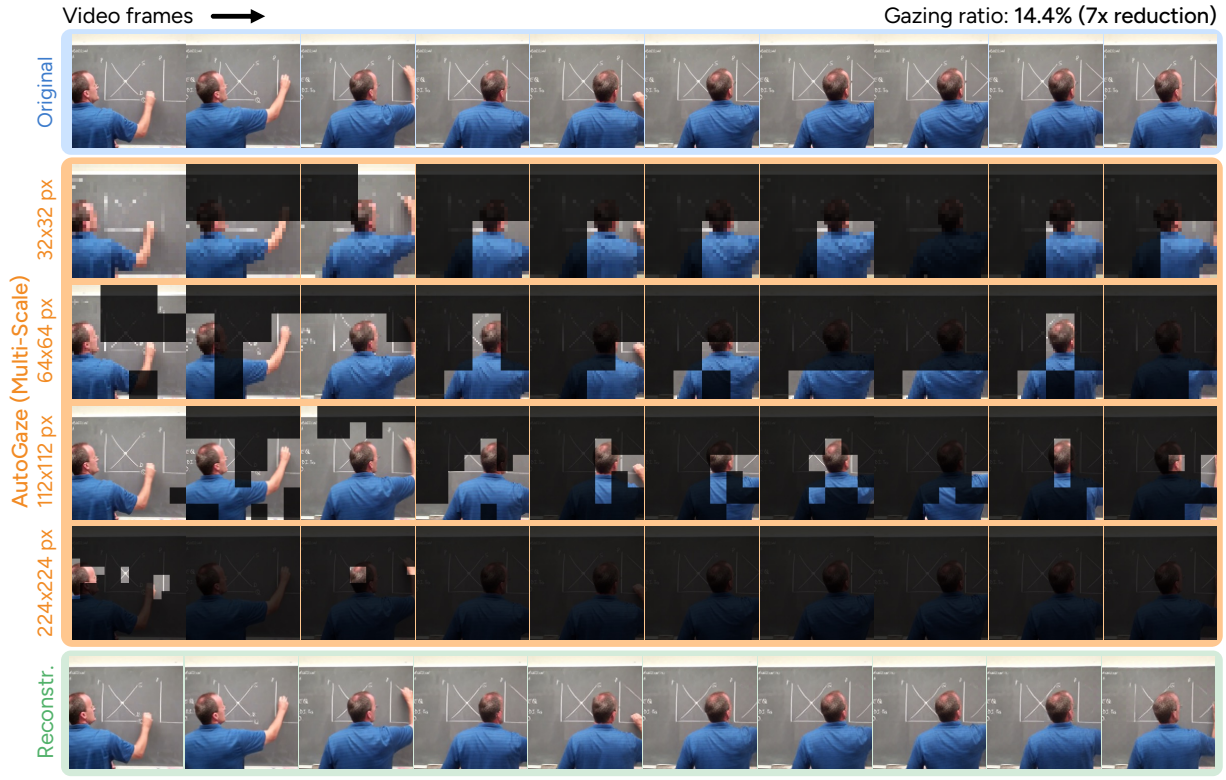

Figure 19. **Chalkboard lecture.** The minimal patches are selected to reconstruct both the lecturer's movement and the chalkboard writing.

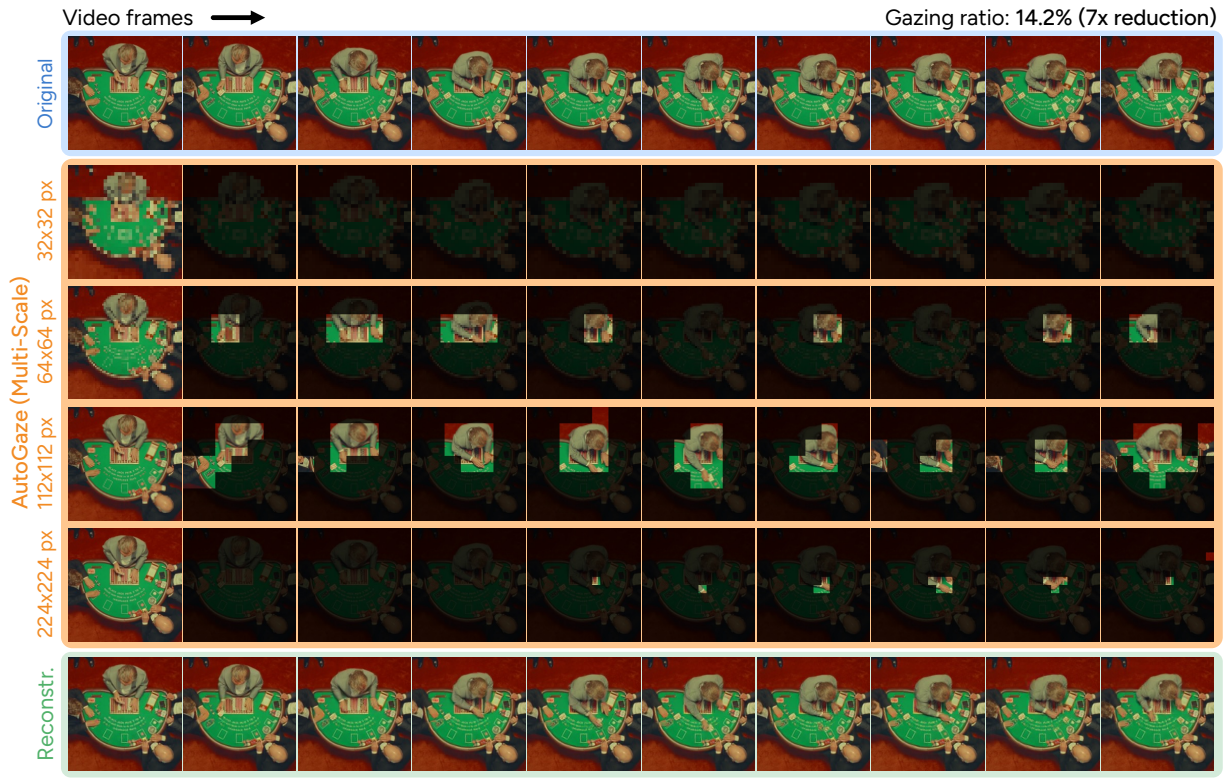

Figure 20. **Film clip.** In this game of Blackjack, AutoGaze uses finer scales to capture hand movements and cards.

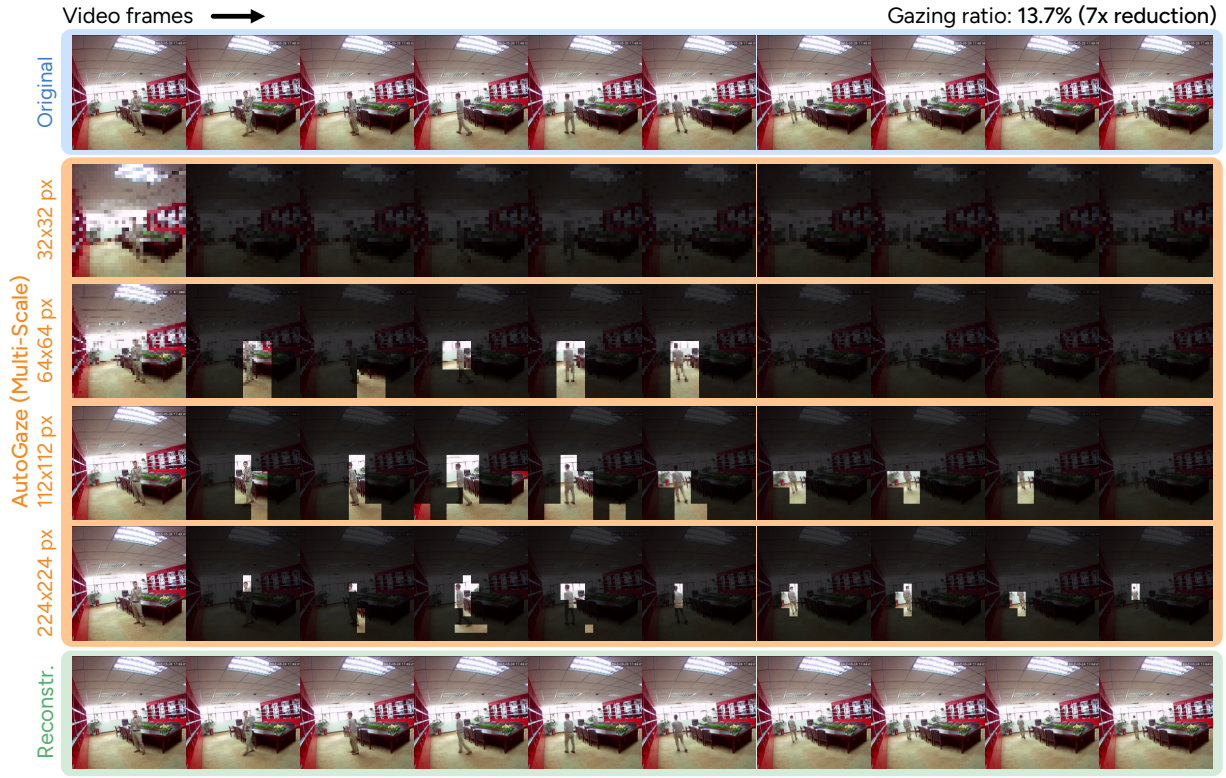

Figure 21. **Fisheye lens.** AutoGaze can select patches that track moving objects with appropriate scales even with lens distortion.

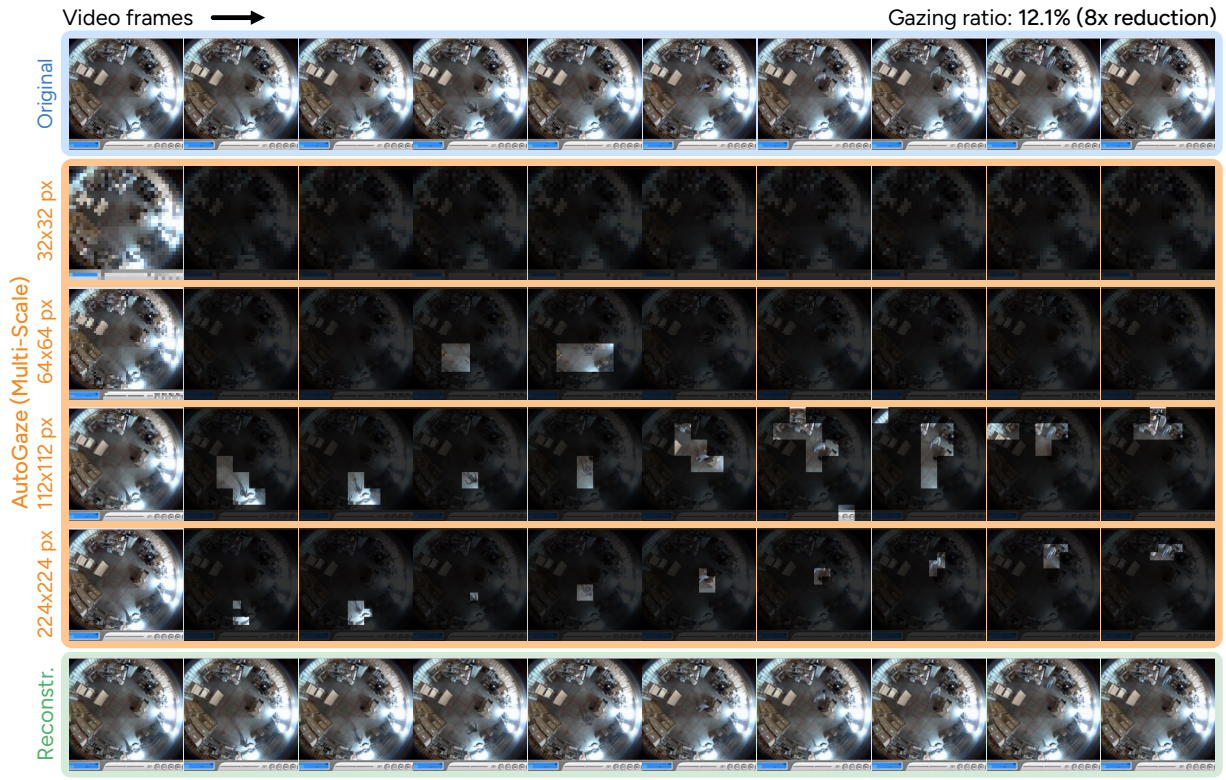

Figure 22. **Fisheye lens.** In this overhead video with lens distortion, AutoGaze tracks the walking person and ignores the static warehouse.

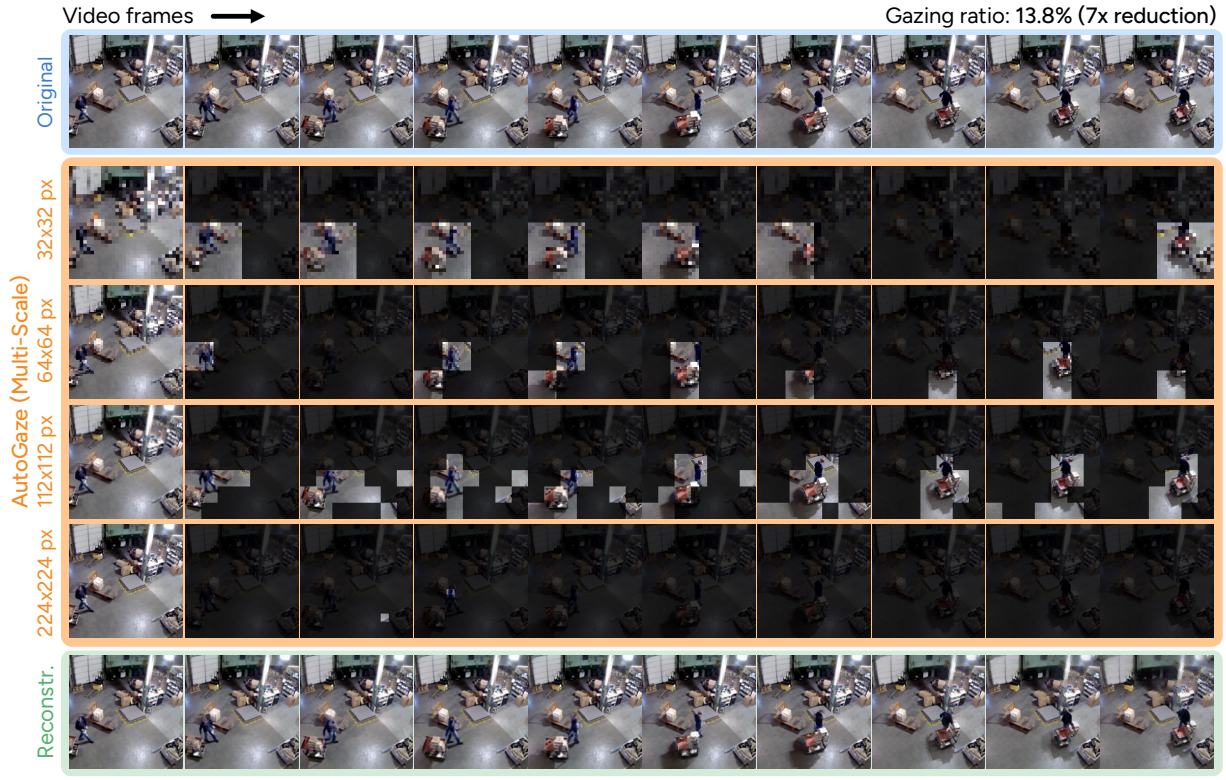

Figure 23. **Warehouse example.** In this video, AutoGaze selects just enough patches to reconstruct the moving person and cart.

Figure 24. **Nighttime driving.** AutoGaze can be used on nighttime videos such as this one, capturing pedestrians and passing cars.

Figure 25. **Robot arm video.** AutoGaze uses different scales to gaze at the robot arm and marker.

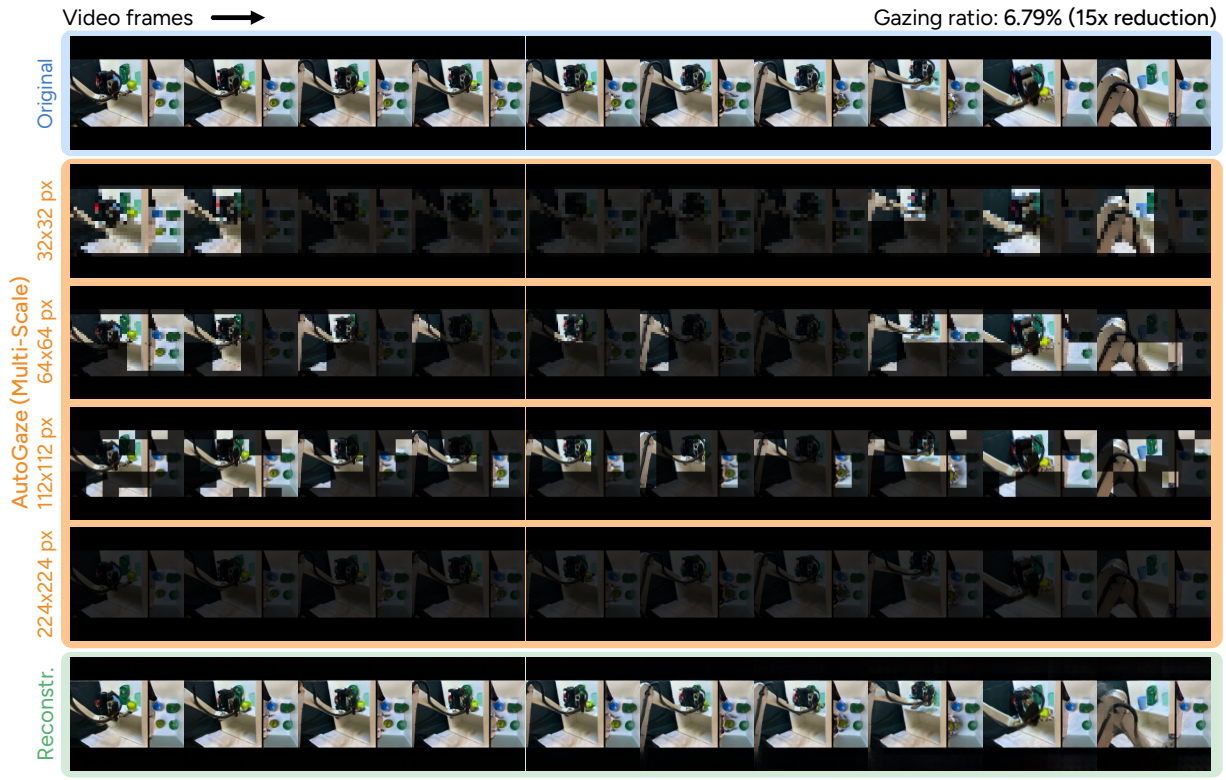

Figure 26. **Multiple perspectives.** Given two side-by-side videos, AutoGaze selects patches in both halves to reconstruct the video.

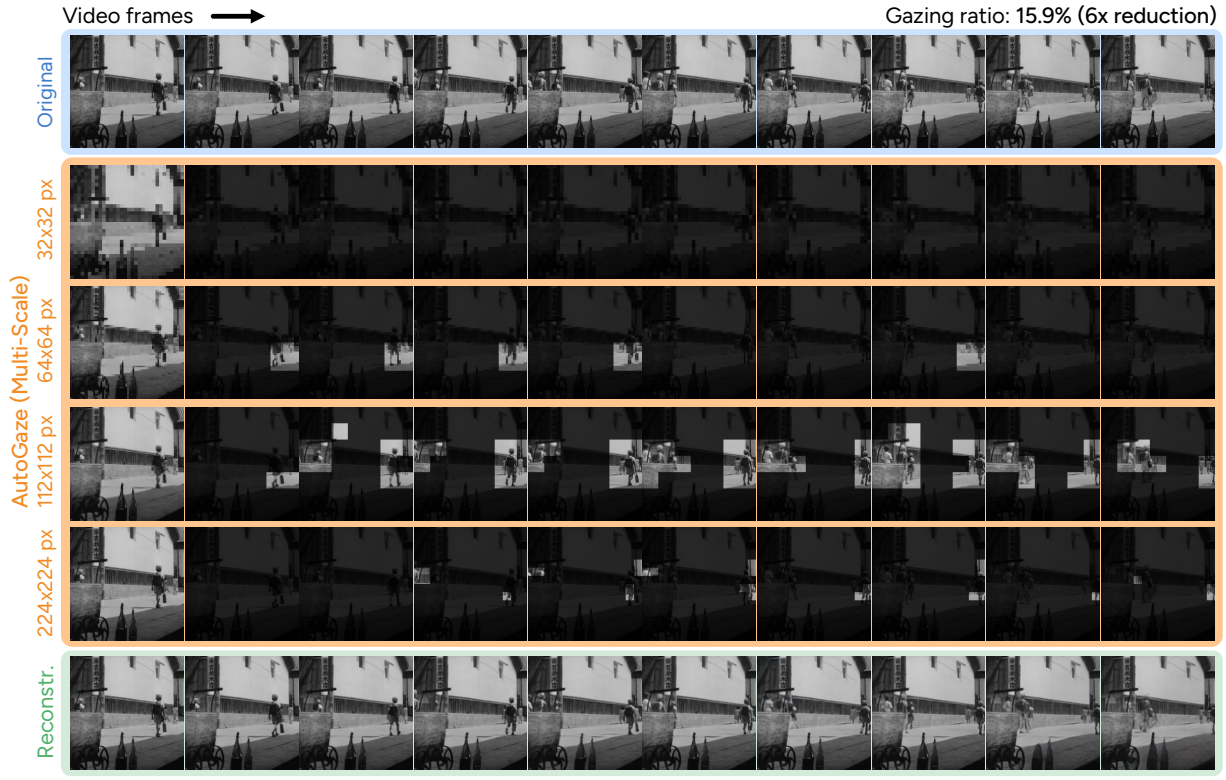

Figure 27. **Black-and-white film.** In this clip where people are walking, AutoGaze uses finer scales to select people as they get smaller.

Figure 28. **Swapping object.** In an OOD scenario where the skateboard swaps out halfway through the video, AutoGaze continues to track the moving parts and properly reconstruct each video frame.

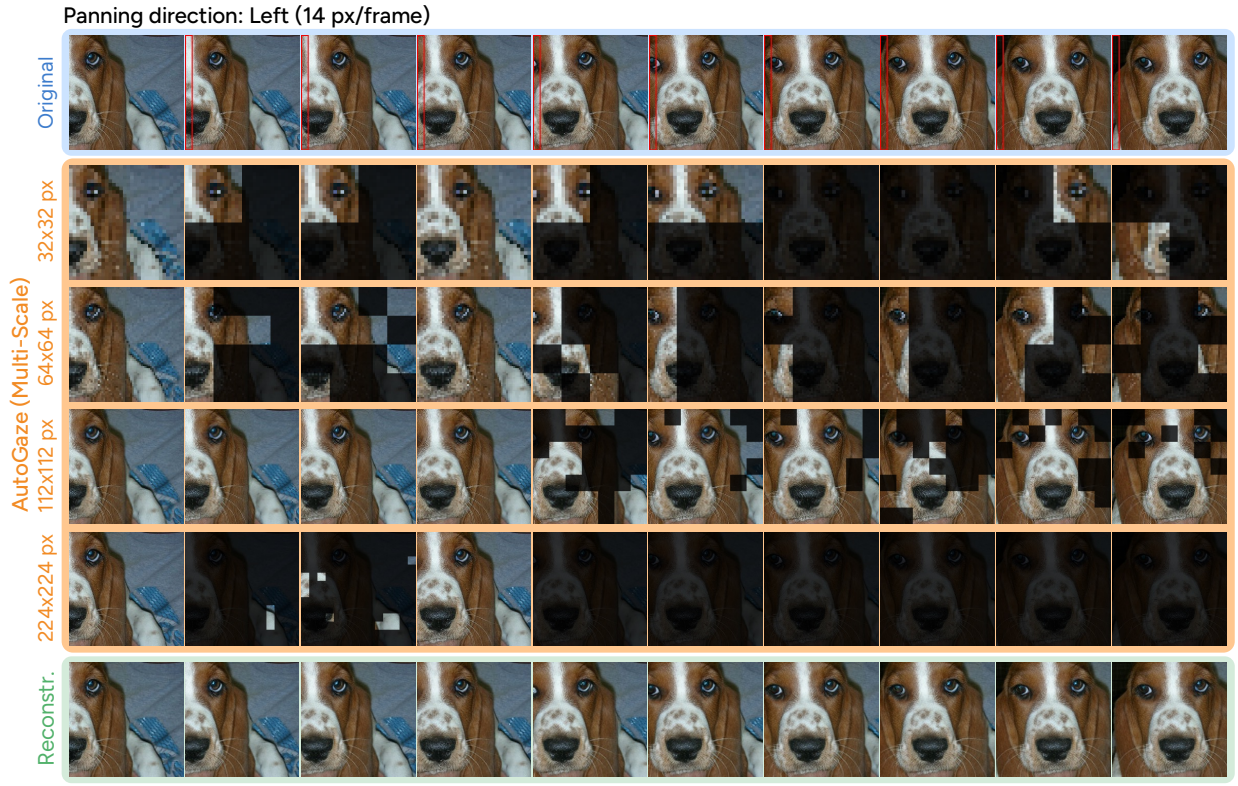

Figure 29. **Panning over a static image.** As the image slides to the left (new pixels highlighted with a red border), AutoGaze does not perfectly track movement; it can select regions that were in different parts of previous frames.

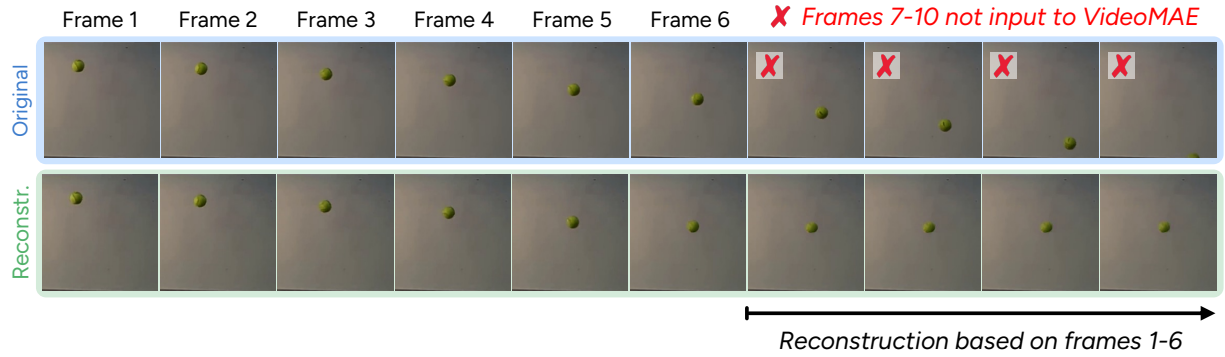

Figure 30. **Predicting next frames from physics knowledge.** Given the first 6 frames of a ball falling along a parabolic path, VideoMAE is unable to reconstruct the next frames to reflect the continued falling motion.