A Architecture, Training, and Evaluation Details

A.1 Video Diffusion Training Details

We use the same base architecture and training setup as [45] which utilizes Video U-Net architecture with 3 residual blocks of 512 base channels and channel multiplier [1, 2, 4], attention resolutions [6, 12, 24], attention head dimension 64, and conditioning embedding dimension 1024. We use noise schedule log SNR with range [-20, 20]. We make modifications Video U-Net to support first-frame conditioning during training. Specifically, we replicate the first frame to be conditioned on at all future frame indices, and apply temporal super resolution model condition on the replicated first frame by concatenating the first frame channel-wise to the noisy data similar to [46]. We use temporal convolutions as opposed to temporal attention to mix frames across time, to maintain local temporal consistency across time, which has also been previously noted in [19]. We train each of our video diffusion models for 2M steps using batch size 2048 with learning rate 1e-4 and 10k linear warmup steps. We use 256 TPU-v4 chips for our first-frame conditioned generation model and temporal super resolution model.

We use T5-XXL [22] to process input prompts which consists of 4.6 billion parameters. For combinatorial and multi-task generalization experiments on simulated robotic manipulation, we train a first-frame conditioned video diffusion models on 10x48x64 videos (skipping every 8 frames) with 1.7B parameters and a temporal super resolution of 20x48x64 (skipping every 4 frames) with 1.7B parameters. The resolution of the videos are chosen so that the objects being manipulated (e.g., blocks being moved around) are clearly visible in the video. For the real world video results, we finetune the 16x40x24 (1.7B), 32x40x24 (1.7B), 32x80x48 (1.4B), and 32x320x192 (1.2B) temporal super resolution models pretrained on the data used by [19].

A.2 Inverse Dynamics Training Details

UniPi's inverse dynamics model is trained to directly predict the 7-dimensional controls of the simulated robot arm from an image observation mean squared error. The inverse dynamics model consists of a 3x3 convolutional layer, 3 layers of 3x3 convolutions with residual connection, a mean-pooling layer across all pixel locations, and an MLP layer of (128, 7) channels to predict the final controls. The inverse dynamics model is trained using the Adam optimizer with gradient norm clipped at 1 and learning rate 1e-4 for a total of 2M steps where linear warmup is applied to the first 10k steps.

A.3 Baselines Training Details

We describe the architecture details of various baselines below. The training details (e.g., learning rate, warm up, gradient clip) of each baseline follow those of the inverse dynamics model detailed above.

Transformer BC [6, 26]. We employ the same transformer architecture as the 10M model of [26] with 4 attention layers of 8 heads each and hidden size 512. We apply 4 layers of 3x3 convolution with residual connection to extract image features, which, together with T5 text embeddings, are used as inputs to the transformer. We additionally experimented with vision transformer style linearization of the image patches similar to [26], but found the performance to be similar. We use a context length of 4 and skip every 4 frames similar to UniPi's inverse dynamics. We tried increasing the context length of the transformer to 8 but it did not help improve performance.

Transformer TT [25]. We use a similar transformer architecture as the Transformer BC baseline detailed above. Instead of predicting the immediate next control in the sequence as in Transformer BC, we predict the next 8 controls (skipping every 4 controls similar to other baselines) at the output layer. We have also tried autoregressively predicting the next 8 controls, but found the errors to accumulate quickly without additional discretization.

State-Based Diffusion [21]. For the state-based diffusion baseline, we use a similar architecture as UniPi's first-frame conditioned video diffusion, where instead of diffusing and generating future image frames, we replicate future controls across different pixel locations and apply the same U-Net structure as UniPi to learn state-based diffusion models.

A.4 Details of the Combinatorial Planning Task

In the combinatorial planning tasks, we sample random 6 DOF poses for blocks, colored bowls, the final placement box. Blocks start off uncolored (white) and must be placed in a bowl to obtain a color. The robot then must manipulate and move the colored block to have the desired geometric relation in the placement box. The underlying action space of the agent corresponds to 6 joint values of robot plus a discrete contact action. When the contact action is active, the nearest block on the table is attached to the robot gripper (where for methods that predict continuous actions, we thresholded action prediction > 0.5 to correspond to contact). Given individual action predictions for different models, we simulate the next state of the environment by running the joint controller in Pybullet to try reach the predicted joint state (with a timeout of 2 seconds due to certain actions being physically infeasible). As only a subset of the video dataset contained action annotations, we trained the inverse-dynamics model on action annotations from 20k generated videos.

A.5 Details of the CLIPort Multi-Environment Task

In the CLIPort environment, we use the same action space as the combinatorial planning tasks and execute actions similarly using the built in joint controller in Pybullet. As our training data, we use a scripted agent on `put-block-in-bowl-unseen-colors`, `packing-unseen-google-objects-seq`, `assembling-kits-seq-unseen-colors`, `stack-block-pyramid-seq-seen-colors`, `tower-of-hanoi-seq-seen-colors`, `assembling-kits-seq-seen-colors`, `tower-of-hanoi-seq-unseen-colors`, `stack-block-pyramid-seq-unseen-colors`, `packing-seen-google-objects-seq`, `packing-boxes-pairs-seen-colors`, `packing-seen-google-objects-group`. As our test data, we used the environments `put-block-in-bowl-seen-colors`, `packing-unseen-google-objects-group`, `packing-boxes-pairs-unseen-colors`. We trained the inverse dynamics on action annotation across the 200k generated videos.

B Additional Results

B.1 Additional Results on Combinatorial Generalization

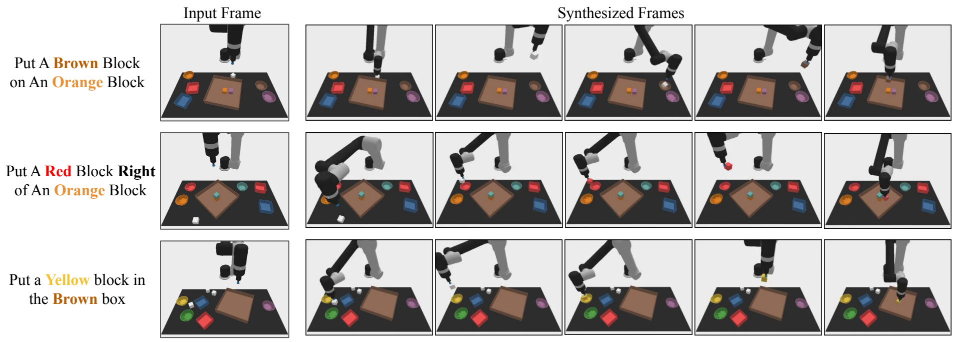

Figure 10: **Combinatorial Video Generation.** Additional results on UniPi's generated videos for unseen language goals at test time.

B.2 Additional Results on Multi-Environment Transfer

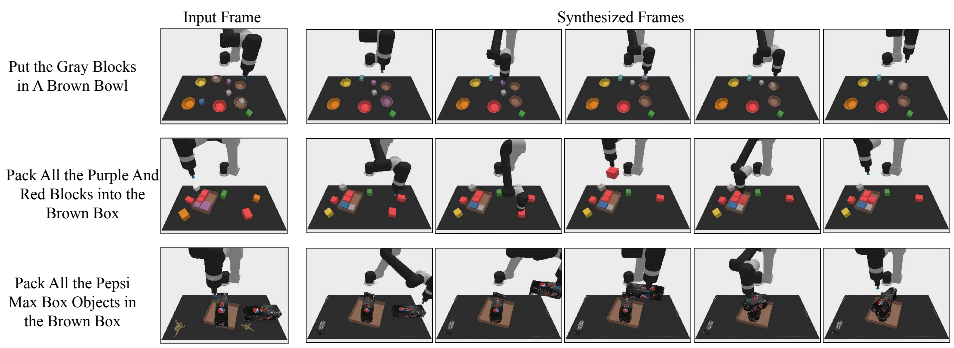

Figure 11: **Multitask Video Generation.** Additional results on UniPi's generated video plans on different new tasks in the multitask setting.

B.3 Additional Results on Real-World Transfer

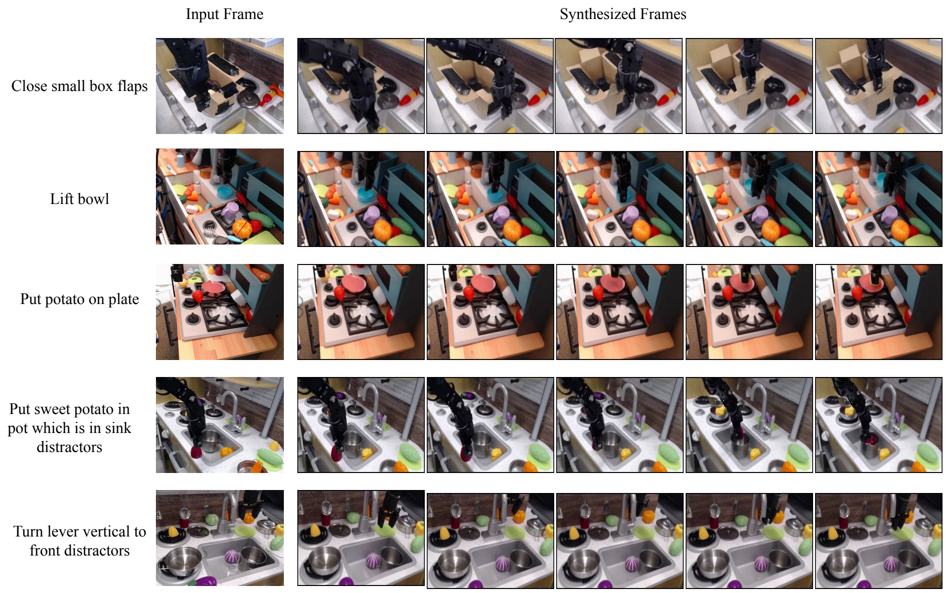

Figure 12: **High Fidelity Plan Generation.** Additional results on UniPi's high resolution video plans across different language prompts.