Imagine2Servo: Intelligent Visual Servoing with Diffusion-Driven Goal Generation for Robotic Tasks

Prepared by:

Mohamed Magdy Atta

Shahid Aahmed Hasib

Mateus Pedrosa

Mohamed Alsisi

May 22, 2025

for

AI Project 2 : Data-Driven Machine Perception

Submitted to:

Prof. Ricard Marxer

Santiago Cuervo

Séverin Baroudi

Contents

1 Introduction & Motivation

It was known for a long time in the robotics field that autonomous robotic systems follows a three-steps architecture that separate the perception, planning, and control during execution as shown in figure (1).

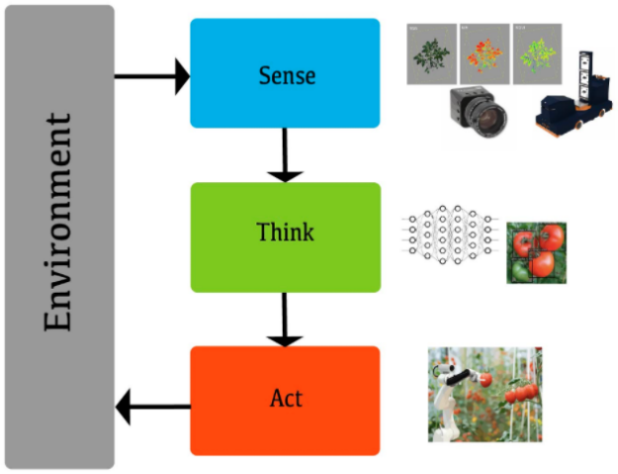

Figure 1: The Traditional Robotic system 3-step architecture

Such traditional architectures usually require a predefined goal location or image and depend on a considerable number of sensors for feedback making them limited in adaptability and scalability. For instance, in a lot of real-world scenarios, robots need to perform tasks in environments where the final goal state is unknown at deployment, or where the robot must interpret high-level instructions rather than simply matching a reference location or image. These constraints hinder the deployment of robots in unseen and dynamic environments.

With the rise of the new deep learning models recently, a new robotic architecture called action models was introduced to overcome the mentioned limitations by leveraging recent advances in generative models, specifically the diffusion-based image editing models. The new aarchitecture merges both perception and planning into a single step as shown in figure (2) to enable robots to perform long-range navigation and manipulation tasks starting from arbitrary initial states without the need for predefined goal locations or images and with a minimal sensor suite, such as a single camera mounted on the robot base or end-effector. This is done by synthesizing intermediate goal images from high-level language prompts and current sensory observations. Overcoming such limitations was our motivation behind choosing that 2-step framework. The specific action model we chose is Imagine2Servo which focuses on performing Visual Servoing intelligently with diffusion-driven goal generation for robotic tasks.

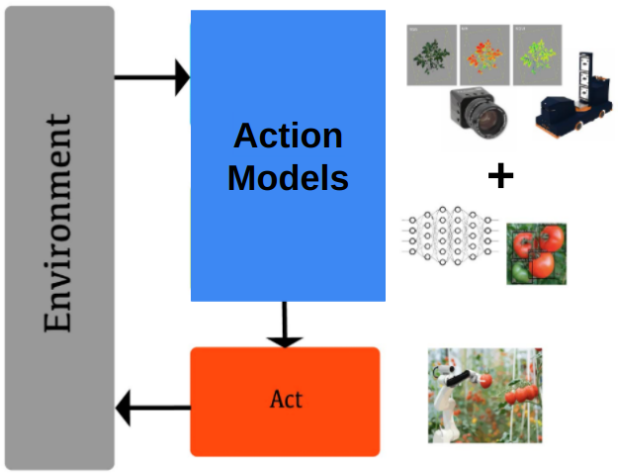

Figure 2: The New Robotic system 2-step architecture

2 Related Work

2.1 Visual servoing

Visual Servoing is a control strategy used in robotics where the robot's movements are directly controlled based on the error between the current image (from a camera) and a target image or position. The primary idea is to minimize the difference between the target object's image or position and its actual position in the camera view. This approach allows the robot to perform tasks like object tracking grasping, and different manipulation tasks solely using visual information from the camera. Visual servoing classical methods such as Image-Based Visual Servoing (IBVS) and Position-Based Visual Servoing (PBVS) require a predefined goal image and substantial visual overlap between the robot's starting and target positions. The Image-Based Visual Servoing control diagram is illustrated below:

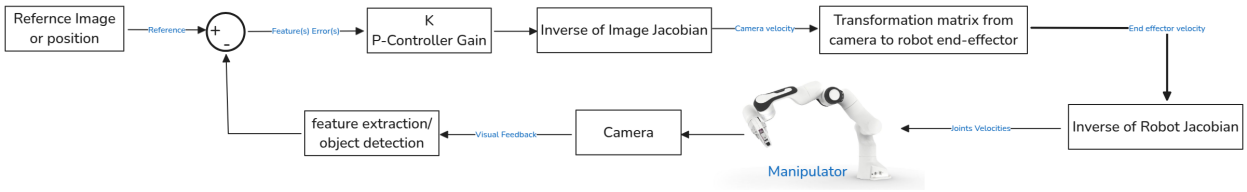

Figure 3: Image-Based Visual Servoing Control Diagram

2.2 Generative Action Models

In the recent literature, there are two paradigms that use generative models for action planning to addresses the challenge of translating high-level commands into executable robotic actions through direct visual annotation or generative visual planning.

One of the examples of visual annotation is the paper called Genima. Its method emphasizes explicit action annotation on images. It is a behavior-cloning agent that fine-tunes Stable Diffusion

to "draw joint-actions" as targets on RGB images. These images are fed into a controller that maps the visual targets into a sequence of joint-positions as shown in figure (4).

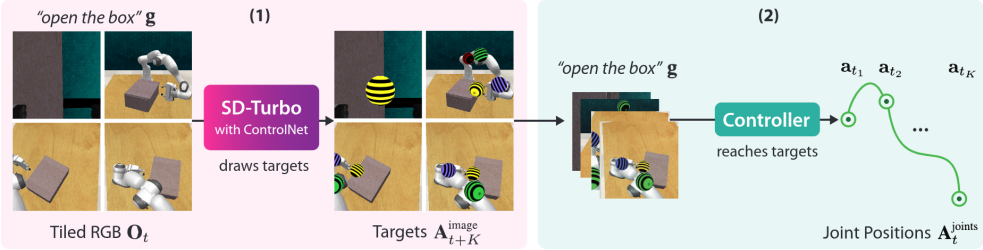

Figure 4: Genima Overview

Given an image-to-image dataset, Genima authors finetune Stable Diffusion with ControlNet to draw targets on observation images. ControlNet is a two-stream architecture: one stream with a frozen Stable Diffusion UNet that gets noisy input and language descriptions, and a trainable second stream that gets a conditioning image to modulate the output. This architecture retains the text-to-image capabilities of Stable Diffusion, while fine-tuning outputs to spatial layouts in the conditioning image. The controller can be implemented with any visuomotor policy that maps RGB observations to joint-positions. Genima authors use ACT – a Transformer-based policy architecture – for its fast inference-speed and training stability.

In Genima paper, authors used multiple cameras fixed in different locations to capture the whole scene continuously. Such a configuration is more common with robots that work at a fixed base position but not desirable for robots that move freely. For this reason, we chose to implement the other option of generating visual planning in another paper called Imagine2Servo which focuses on synthesizing visual outcomes from language prompts and current camera observations.

Imagine2Servo paper deals with the task of generating the next subgoal for the servoing controller as editing the pixels of the current input image. Given the current image and task description P, authors aim to generate the subgoal image using the foresight diffusion model. They employ RTVS (Real-Time Visual Servoing) as the Image-Based Visual Servoing (IBVS) controller to reach the subgoals predicted by the foresight module. They use the RTVS servoing algorithm without any fine-tuning to the newer environments as shown in figure (5).

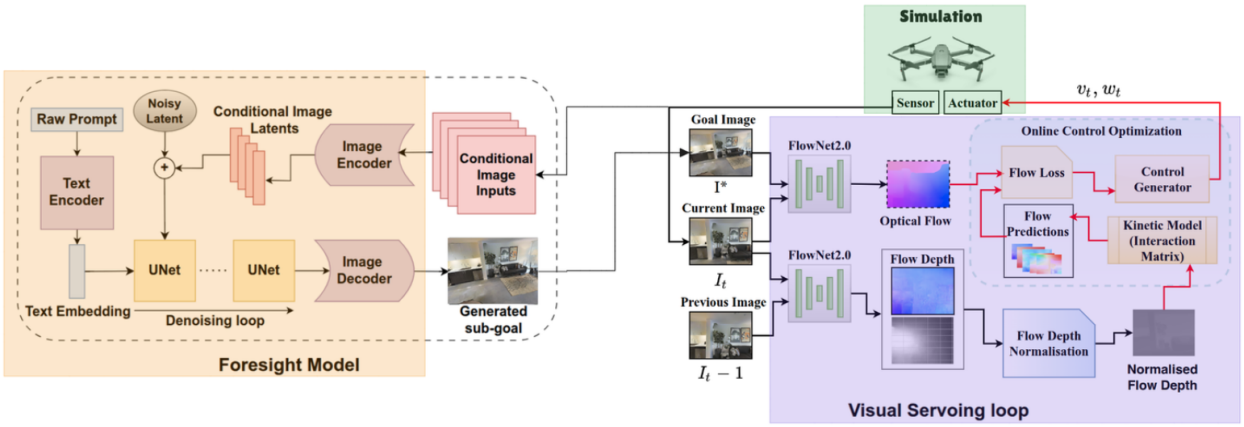

Figure 5: Imagine2Servo architecture

The Foresight Model serves as the generative part of the framework, producing sub-goal images that guide the robot to complete high-level tasks specified by language prompts. The process begins with a raw text prompt, such as ”reach the red ball” which is transformed into a dense semantic embedding by a text encoder. This embedding, combined with a noisy latent vector and conditional image latents representing the robot's current observation, forms the input to a denoising diffusion process implemented via the U-Net architecture. The U-Net iteratively refines the noisy latent to generate a sub-goal image. Finally, an image decoder translates the refined latent into an actual image, representing the next intermediate state the robot should achieve to complete its the task. The authors used Instructpix2pix image editing framework in this step.

The Visual servoing loop converts the generated sub-goal image into actionable control commands. This loop receives the sub-goal (goal image), the robot's current camera observation, and the previous image as inputs. Using FlowNet2.0, it estimates the optical flow between the goal and current images, determining the pixel-wise motion required to move toward the sub-goal.

The online control optimization inside the visual servoing computing optimal control commands based on visual feedback. The simulation block provides sensory data and accepts actuator commands, creating a feedback loop. Online control optimization uses the difference between predicted and actual flow and flow predictions to compute the optimal velocity commands, which are sent into the simulation. The control generator optimizes the flow loss to generate control commands. The authors used the RTVS method in this step.

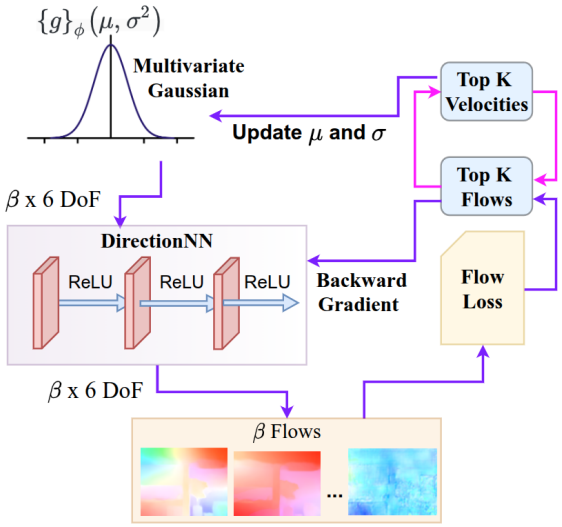

Figure 6: Control Generator

As shown in figure (6), authors sample a vector from a multivariate Gaussian distribution and pass it through a directional neural network which is trained online during each MPC step, allowing it to quickly learn small pose changes and retain weights across iterations, which improves servoing efficiency by reducing the required optimization steps. The kinetic model generates pseudo-flows and the flow loss is computed against the target flow. In each step, they pick the top velocities corresponding to the least flow errors to update the Gaussian parameters, while the mean velocities is sent to the actuators.

The control generation architecture solve the high-dimensional problem inherent in visual servoing. Traditional Model Predictive Control (MPC) methods struggle with such complexity. This

approach leverages the differentiable cross-entropy method (DCEM) to enable adaptive sampling of control commands over multiple MPC steps. Unlike classical CEM, DCEM updates the gaussian sampling parameters iteratively, preserving memory of effective samples and aligning optimization directly with the downstream task objective, thereby reducing the number of MPC iterations and servoing time.

The generated velocities effectiveness is evaluated using pixel-wise flow error computed between the current and goal images. This iterative process continues until convergence, producing velocity commands to guide the robot efficiently toward its goal while minimizing flow error.

2.3 Image Editing Models

The specific model used for the foresight in the architecture is called InstructPix2Pix. It enables intuitive image editing through natural language instructions. Given any input image and a written instruction (such as "replace the fruits with cake" or "make his jacket out of leather") the model automatically edits the image to match the instruction, without needing extra example images, masks, or detailed descriptions. The system was trained by combining the strengths of two powerful AI models: a language model, which generates editing instructions and captions, and a text-to-image model, which creates pairs of before-and-after images based on these captions. Using this synthetic data, InstructPix2Pix learns to directly perform image edits in a single forward pass, making it fast and responsive. At inference time, it generalizes to real images and arbitrary, user-written instructions, allowing the user to make creative and precise edits simply by telling the model what to do in plain language.

In our system, given the current frame from the manipulator's camera and a text prompt with the task instruction, InstructPix2Pix is responsible for generating an edited version of the original image, correspoding to the next subgoal for the manipulator.

3 Our Implementation & Key Contributions

In our implementation of the Imagine2Servo framework, we began by designing the simulation setup using PyBullet simulator, where a Franka Emika Panda 7-DOF manipulator equipped with a single eye-in-hand camera was used to perform a range of reaching tasks. To support robust training and evaluation, we generated a comprehensive dataset consisting of variations of reaching tasks such as "reach the red ball" and "reach the blue box" with each variation containing a number of demonstrations and five camera frames recorded per demonstration as ground truth sub goals with randomized initial frames to ensure diversity. A key contribution in our pipeline was the replacement of the traditional FlowNet2 optical flow network with NeuFlow2, a more performant model on edge devices, to improve the accuracy and stability of visual servoing in the RTVS loop. Implementation of the diffusion-driven goal generation pipeline, as described in the original paper, was done. This pipeline removes the dependency on predefined goal images and allows for more adaptive and language-driven robotic behavior. We fine-tuned the InstructPix2Pix diffusion model on our custom dataset, focusing on single-view camera input to mimic realistic robotic perception scenarios. The entire system is fully integrated, including simulation, diffusion model, and a closed-loop visual servoing loop, resulting in a reproducible and extensible architecture. To facilitate further research and benchmarking, we have made all trained models and pipeline codes

publicly available, supporting reproducibility and community-driven advancement in intelligent visual servoing. The modified architecture is shown in Figure 7.

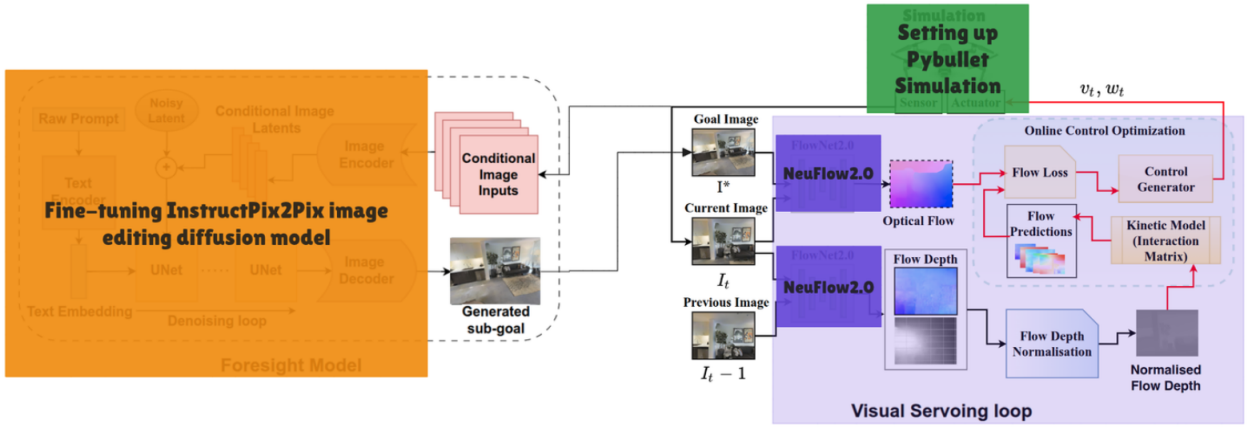

Figure 7: The Implemented Imagine2Servo Modified Architecture

4 Experiments & Results

4.1 Synthetic Dataset Generation

To generate the dataset, we used the PyBullet simulator with a Franka Emika Panda 7-DOF manipulator and one camera placed in its end effector. We performed 4 variations of the reaching task and for each variation we created a text prompt: "reach the red ball", "the blue ball", "the red box", and "the blue box". For each prompt, we recorded 250 demos of the manipulator performing the task, consisting of 5 frames along the trajectory of the task, which represent the subgoals. To finetune the model, we used pairs of consecutive frames as the input image and ground truth, respectively. We also generated a more simpler dataset with only one reaching command at beginning of our training.

At the beginning of each demo, a script would randomize the initial conditions, changing the placement of objects, its colors, shape, and number of objects, so that the model would learn to generalize to unseen scenarios.

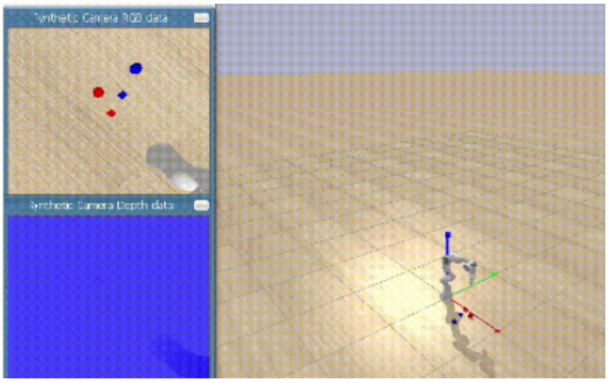

Figure 8: Dataset Generation in Pybullet Simulator

4.2 Fine-Tuning

To fit the InstructPix2Pix model for our robotic manipulation scenario, we designed a fine-tuning process centred on a small but meaningful dataset. This dataset was composed of 500 distinct manipulation demonstrations, with each demonstration represented by five frames that captured different stages of a specific task. This frame-based approach not only introduced temporal diversity but also mimicked real-world robotic interaction more closely.

The model was initialised using the publicly available `(timbrooks/instruct-pix2pix)` weights from the Hugging Face Hub. This checkpoint provided a solid starting point, pre-trained on a broad set of image editing tasks. Our goal was to adapt it for domain-specific visual transformations based on natural language instructions. Fine-tuning was conducted using the `Diffusers` library, which offered built-in support for Stable Diffusion pipelines, allowing for easy adaptation of both training and inference components.

A significant step in this pipeline was modifying the dataset loader. Unlike the original setup, which relied on a standard dataset interface, our approach required parsing a custom `metadata.json` file. Each entry in this file describes a sample using three components: the original image, the edited target image, and the corresponding textual prompt.

We adjusted the training script `train_instruct_pix2pix.py` to accept custom keys for these fields using `--original_image_column`, `--edited_image_column`, and `--edit_prompt_column` arguments.

The training was executed on a high-performance computing cluster, utilizing an A100 GPU to manage memory-intensive operations such as mixed-precision training and gradient checkpointing. We employed a batch size of four and trained for 30,000 steps with a learning rate of 5×10^{-5}. To prevent unnecessary memory usage, checkpoints were saved periodically, with only the most recent two retained. During the process, `Weights & Biases (W&B)` was used for real-time tracking of metrics like stepwise loss and convergence patterns.

The training run demonstrated stable loss curves over time, with a sharp decrease in the early stages and consistent convergence by the final steps. This behaviour indicated that the model effectively adapted to the manipulation-focused visual domain. The final model checkpoint was saved at step 30,000 and has been published under the name `shahidhasib1014/instruct-pix2pix-custom` on the Hugging Face Hub.

To gradually improve the model's ability to handle variation in object appearance, we structured the training process in two phases. Initially, we trained the model using a dataset of 250 demonstrations, where each demonstration included five sequential image frames of red-colored target ball. This early run helped the model grasp the structure of the tasks and learn how to interpret edit instructions in a controlled visual setting. However, we noticed that the model's performance was limited when exposed to shapes of different colors. In response, we expanded the dataset to 1000 demonstrations and introduced blue shapes alongside the original red ones. This additional visual diversity allowed the model to learn more generalizable features and respond more effectively to a wider range of prompts during inference.

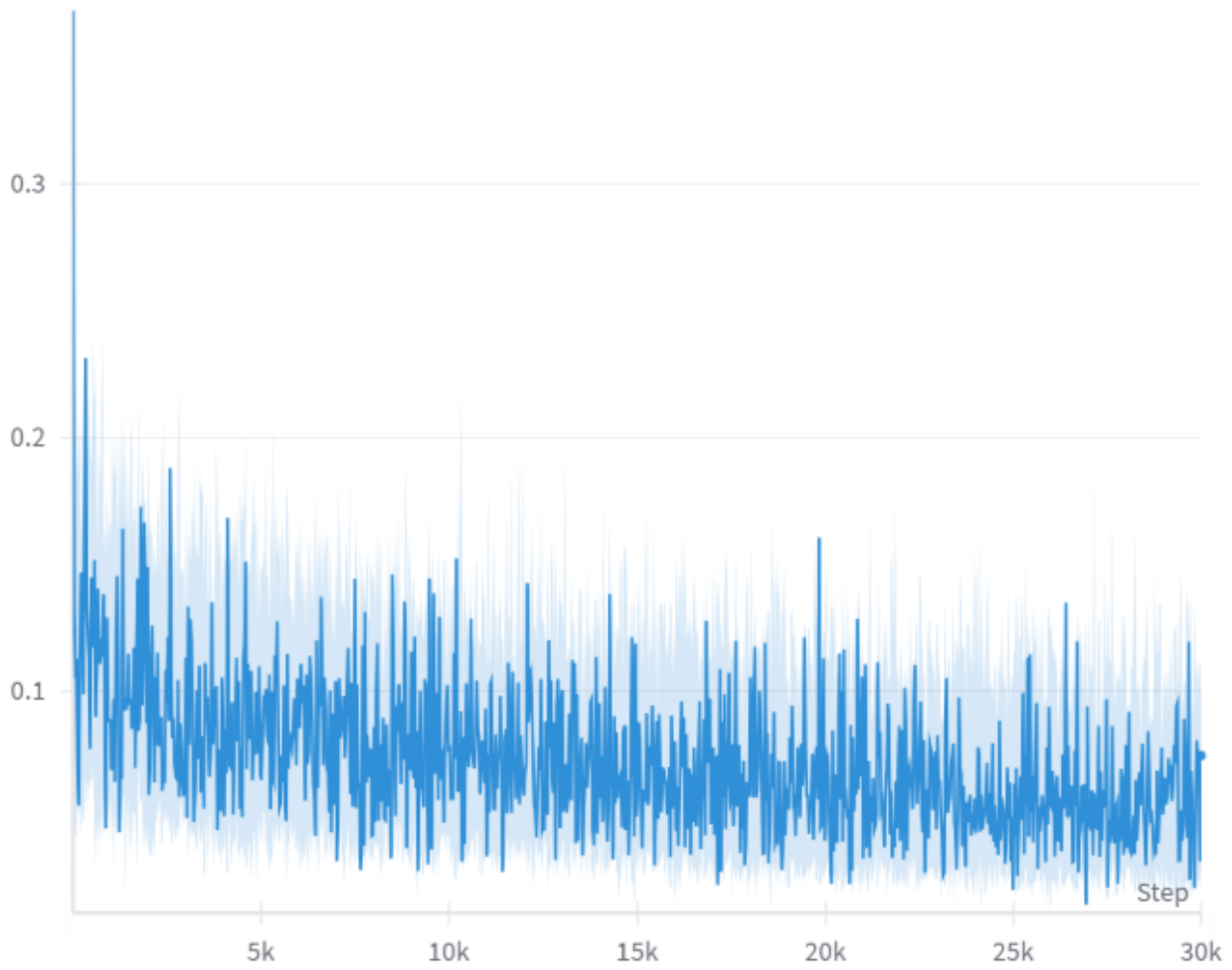

Figure 9: Training Loss

This model is publicly accessible and can be reused or fine-tuned further by others working on similar vision-based robotic applications. Qualitative results confirm that the fine-tuned model is capable of translating visual prompts into realistic and semantically accurate edits, offering valuable cues for downstream robotics tasks such as perception or planning.

4.3 Modified System Integration & Simulation

We developed a simulation and control loop that combines real-time image acquisition, diffusion-based goal prediction, and closed-loop visual servoing. The simulation environment is built on PyBullet, where a Franka Emika Panda robotic arm operates with an eye-in-hand camera. At each iteration, the system captures the current camera frame, processes it, and resizes it to 256×256 pixels for compatibility with the InstructPix2Pix diffusion model. The model, conditioned on both the current image and a task prompt (for example "reach the red ball"), generates a predicted goal image representing the next subgoal for the robot. This predicted image replaces traditional object detection and segmentation modules, providing a prompt-driven target for the servoing controller.

The modified Real-Time Visual Servoing (RTVS) controller then computes the photometric error between the current and predicted goal images and iteratively generates velocity commands to minimize this error. These commands are transformed from the camera frame to the robot's end-effector frame using the current pose and the Jacobian, and are applied as joint velocity controls in the PyBullet simulation. The loop continues until the photometric error is below a predefined threshold or a maximum number of steps is reached. Below is a pseudocode representation of our main integration loop:

Algorithm 1 Our Modified System Integration and Simulation Loop

$i \leftarrow 0$
while goal not reached **do**
    Simulate physics step in PyBullet
    Acquire current camera frame, preprocess and resize to 256×256
    Predict next goal image using InstructPix2Pix with current image and prompt
    Display current and predicted goal images
    Compute initial photometric error between current and goal images
    Initialize RTVS controller with goal image
    while photometric error $>$ threshold **and** step $<$ max steps **do**
        Compute velocity command with the Modified RTVS
        Transform velocity to end-effector frame using current pose and Jacobian
        Apply joint velocity commands in simulation
        Step simulation
        Update photometric error
        Acquire new camera frame
        Increment step counter
    end while
    Increment i
end while

5 Results

5.1 Qualitative Results

Prompt: reach the blue ball

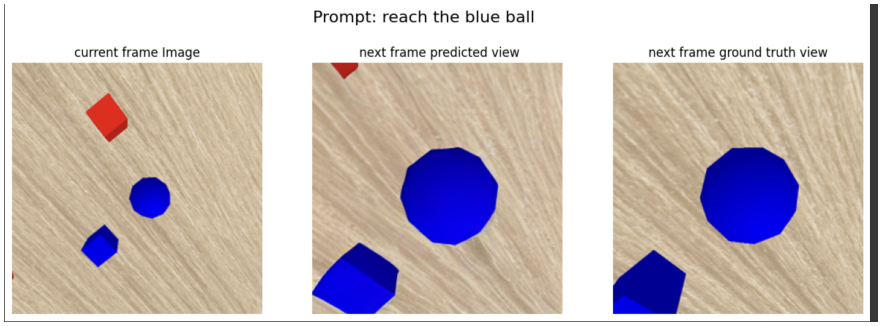

Figure 10: Current frame, predicted frame, and ground truth for the prompt ”reach the blue ball”.

Description: The predicted view closely matches the ground truth, shoe the model's ability to accurately anticipate the movement required to reach the target blue ball between objects.

Prompt: reach the red box

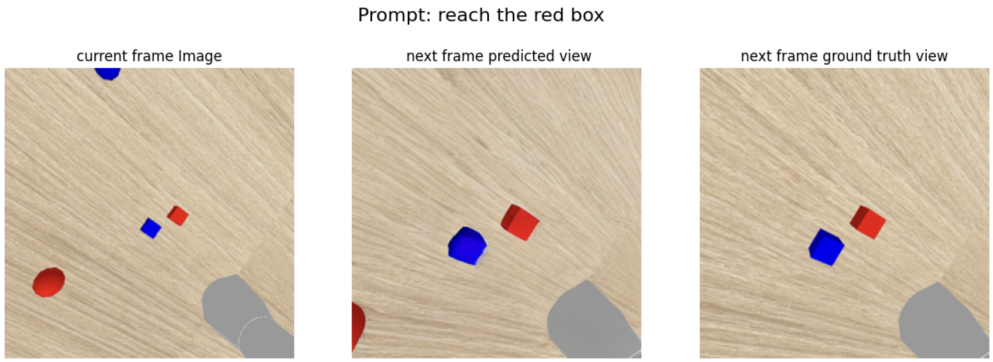

Figure 11: Current frame, predicted frame, and ground truth for the prompt ”reach the red box”.

Description: In this scenario, the environment includes multiple objects. The model successfully predicts the agent's movement toward the red box, as specified in the prompt, and the predicted frame aligns closely with the ground truth.

Prompt: reach the red box

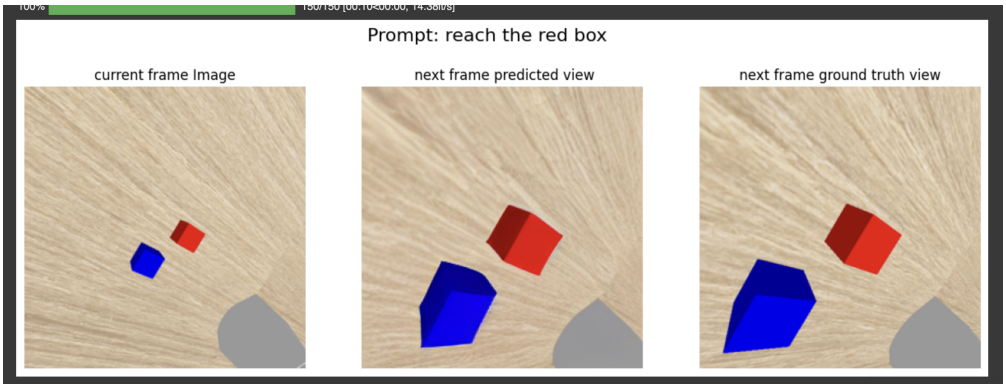

Figure 12: Current frame, predicted frame, and ground truth for the prompt ”reach the red box”.

Description: the model able to interpret the instruction ”reach the red box.” again in sequence frames, showing the agent moving toward the red box in sequence prompts

Prompt: reach the blue ball

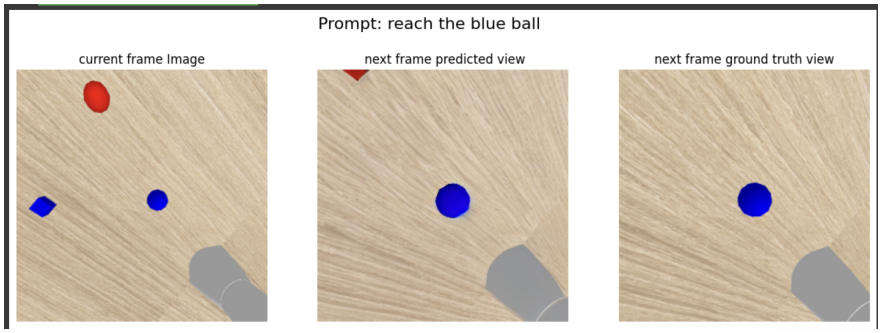

Figure 13: Current frame, predicted frame, and ground truth for the prompt ”reach the blue ball”.

Description: For the prompt ”reach the blue ball,” the model’s prediction (center) shows the agent moving toward the blue ball, and the predicted frame is visually consistent with the ground truth.

Limitations

Prompt: reach the red ball

Description: In response to the prompt ”reach the red ball,” the model accurately predicts the agent’s movement toward the red ball, but he change the ball next to it to box which wrong.

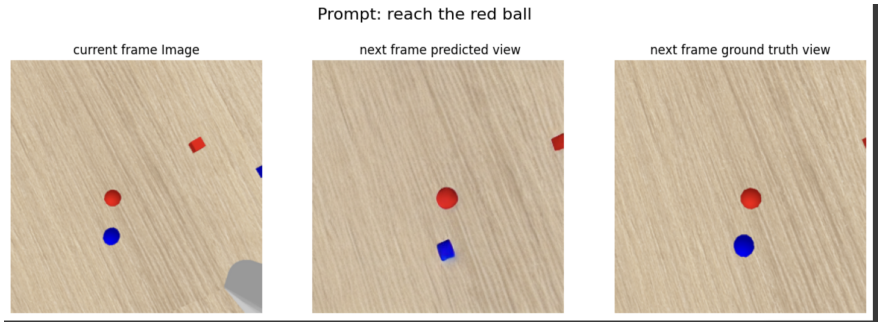

Figure 14: Current frame, predicted frame, and ground truth for the prompt ”reach the red ball”.

Prompt: reach the red ball

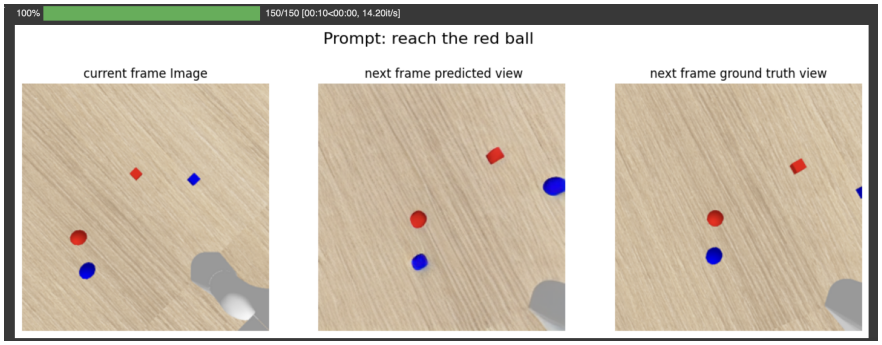

Figure 15: predicted frame, and ground truth for the prompt ”reach the red ball”.

Description: The close alignment between prediction and ground truth highlights the model’s spatial reasoning and goal-directed planning.

Prompt: reach

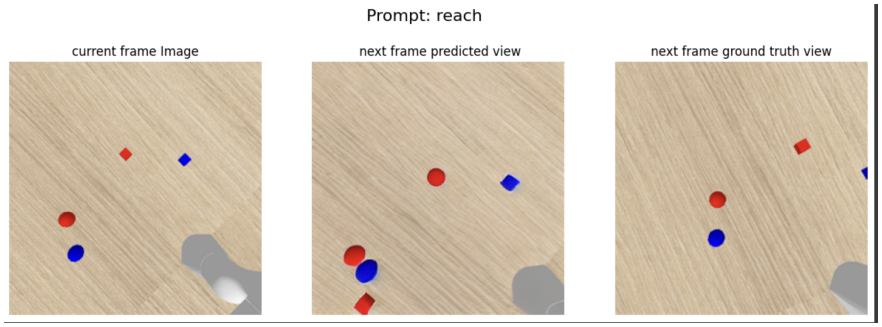

Figure 16: Current frame, predicted frame, and ground truth for the prompt ”reach”.

Description: Here, the prompt is simply ”reach,” without specifying a color. The model predicts a movement toward the nearest object, as seen in the predicted view. The result demonstrates the model’s generalization ability is weak and this one of his limitations.

Prompt:

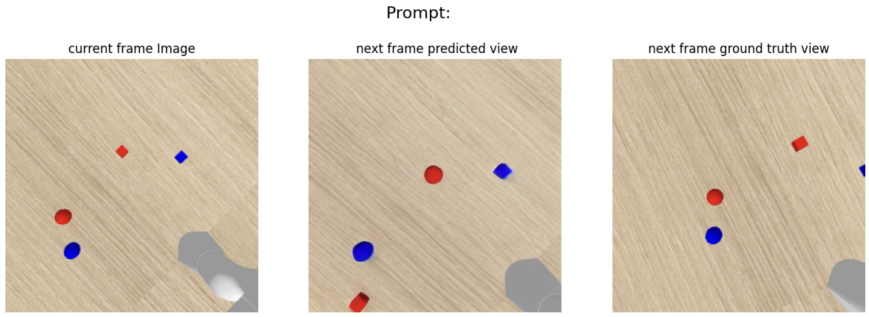

Figure 17: Current frame, predicted frame, and ground truth for an unspecified prompt.

Description: In this example, the prompt is left blank. The model's predicted frame suggests a default reaching action, possibly toward the nearest object. The qualitative similarity between the predicted and actual next frames is not good because he change some object colors.

5.2 Evaluation Results

Across 20 qualitative test trials, the model exhibited a range of behaviors in response to the given prompts. As shown in Table 1, the agent successfully reached the intended object in 11 out of 20 trials, demonstrating the model's core capability to interpret and act on instructions accurately. In 4 trials, the agent reached the correct object, but the model altered the shape or color of surrounding objects, indicating partial success with some visual inconsistencies. In the remaining 5 trials, the model generated a new or unintended object, reflecting a failure mode where hallucinated elements appeared in the scene.

These results suggest that, while the model often performs the intended action, it struggles with maintaining scene consistency and avoiding the creation of extraneous objects. To address these limitations, it is recommended to increase the number of frames used during training (for example, from 5 to 10 frames) and to expose the model to a broader variety of scenarios. This approach is expected to improve both the temporal understanding and generalization ability of the model.

Table 1: Summary of Qualitative Trials

Outcome	**Number of Trials (out of 20)**
Correctly reached the object	11
Reached object, but changed shape/color of surroundings	4
Created a new/unintended object	5

5.3 Simulation Results

We evaluated our complete pipeline implementation in simulation using single object scenes and the robot converged well to the desired object of interest, but took more time than usual due to local minima existence. The following sequence of ordered output frames from simulation testing show the results.

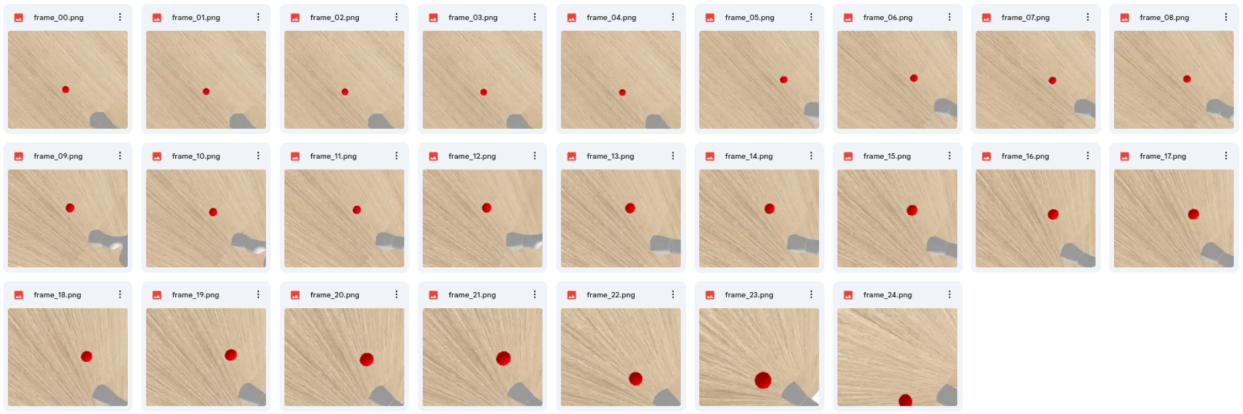

Figure 18: Single object reaching environment results

Next, we evaluated the pipeline in multi-object environment scenes. The results accuracy degraded compared to the single object scenarios, thus indicating the model need more training and increase in number of frames per sample for better performance during inference. The following sequence of ordered output frames from simulation testing show the results.

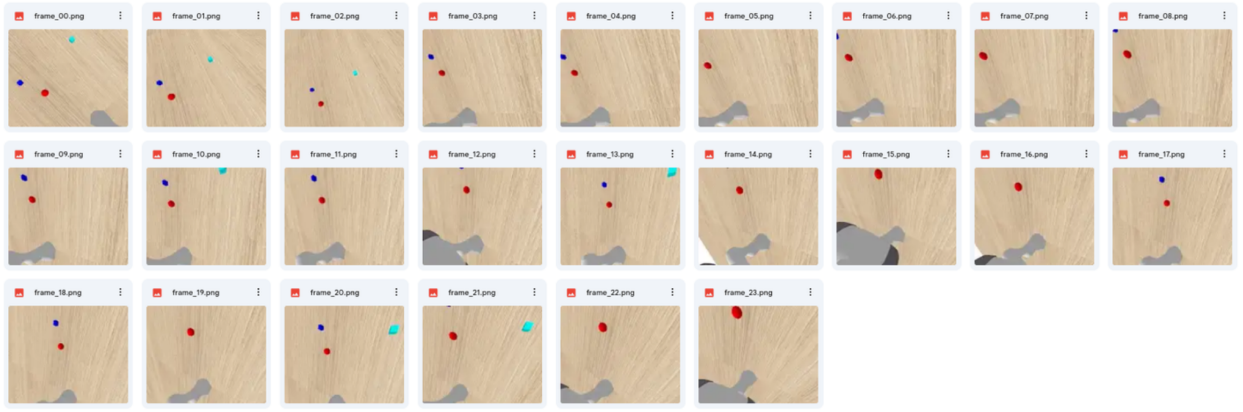

Figure 19: Multi-object reaching environment results

6 Future Work

A number of future directions can further enhance the impact of the Imagine2Servo framework with respect to underwater robotics. First, we plan to increase the the number of frames captured per sample in our dataset to test the effect on the diffusion predictions accuracy during inference. Second, benchmarking the system against a range of different base diffusion models to understand more deeply the generalization performance across different architectures. Third, expanding the simulation and real-world environments to include more complex and diverse tasks to test the system's durability under challenging conditions.

Additionally, we plan to improve to the control generation architecture by incorporating underwater and kinematics constraints as well as multi sensory inputs to extend the applicability of Imagine2Servo to new domains and operational contexts. We also plan to design a controller architecture that can work with low quality and partially wrong diffusion predictions, thus enabling the system to operate even with the limitations of the image editing diffusion models.

7 References

References

[1] P. Pathre, G. Gupta, M. N. Qureshi, M. Brunda, S. Brahmbhatt, and K. M. Krishna, "Imagine2servo: Intelligent visual servoing with diffusion-driven goal generation for robotic tasks," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 13 466–13 472.

[2] M. Shridhar, Y. L. Lo, and S. James, "Generative image as action models," 2024. [Online]. Available: https://arxiv.org/abs/2407.07875

[3] M. N. Qureshi, P. Katara, A. Gupta, H. Pandya, Y. V. S. Harish, A. Sanchawala, G. Kumar, B. Bhowmick, and K. M. Krishna, "Rtvs: A lightweight differentiable mpc framework for real-time visual servoing," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3798–3805.

[4] Z. Zhang, A. Gupta, H. Jiang, and H. Singh, "Neuflow v2: High-efficiency optical flow estimation on edge devices," 2024. [Online]. Available: https://arxiv.org/abs/2408.10161

[5] T. Brooks, A. Holynski, and A. A. Efros, "Instructpix2pix: Learning to follow image editing instructions," in *CVPR*, 2023.

8 Appendix

We uploaded all the code, datasets, and results we developed and integrated to a github repository: Github Link