MIRA: Mental Imagery for Robotic Affordances Supplementary Material

Lin Yen-Chen[1], **Pete Florence**[2], **Andy Zeng**[2], **Jonathan T. Barron**[2],
Yilun Du[1], **Wei-Chiu Ma**[1], **Anthony Simeonov**[1], **Alberto Rodriguez Garcia**[1], **Phillip Isola**[1]
[1]MIT [2]Google

A Additional Real-world Results

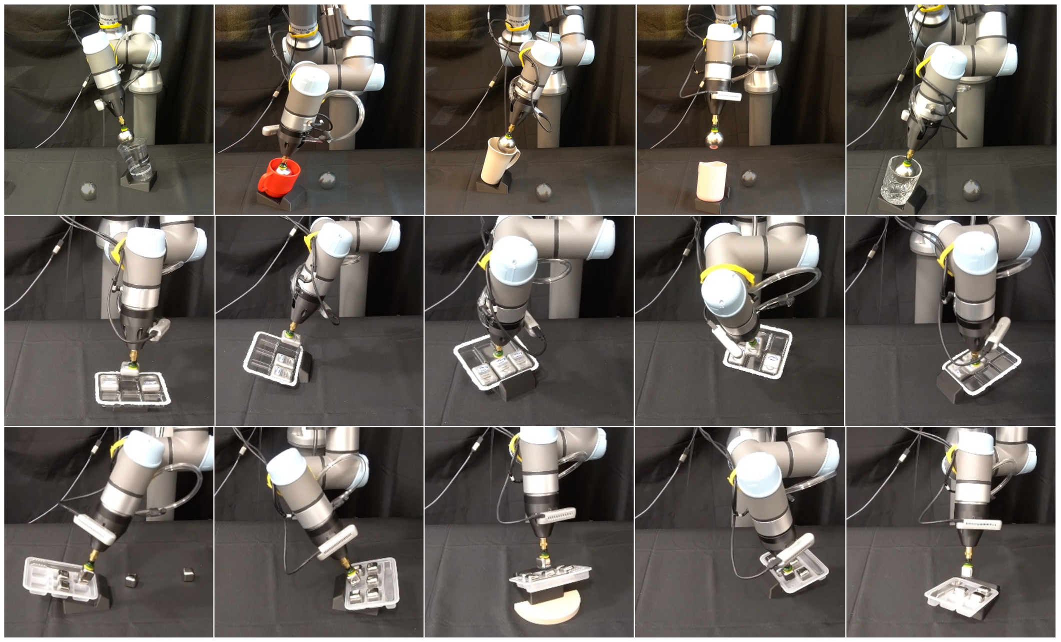

Figure 1: Additional Real-world Qualitative Results. Top-row: placing a metal sphere into cups; middle-row: packing flosses; bottom-row: packing metal cubes.

We present qualitative results in Fig. 1 and recommend readers watch the supplementary video for more comprehensive qualitative results. All models are trained with 10 demonstrations performed on the real robot. For each task, we test our model with 10 random configurations of the environment and present the quantitative results in Table 1. placing-the-metal-sphere-into-cups is trained and evaluated on the same metal sphere and 5 different cups. packing-flosses and packing-metal-cubes are trained and evaluated using the same kits while different numbers of objects are placed on the table or stored in the container.

Task	# Train (Samples)	# Test	Succ. %
placing-the-metal-sphere-into-cups	10	10	80.0
packing-flosses	10	10	60.0
packing-metal-cubes	10	10	70.0

Table 1: Success rates (%) of MIRA trained and evaluated on 3 real-world tasks. Model weights are not shared.

A.1 Examples of Failures

We discuss the representative failures for each task. For placing-the-metal-sphere-into-cups, our model sometimes predicts the wrong pick location and thus fails to pick up the sphere. Also, when the cups are too tall (e.g., goblet), the camera mounted on the robot fails to capture multi-view RGB images that always contain the object, which results in poor NeRF reconstruction. For packing-flosses, NeRF's predicted depths are sometimes incorrect when the empty slot is not supported by the black stand. This will cause the end-effector to go "too deep" toward the container. For packing-metal-cubes, our model sometimes predicts the wrong picking location and fails to pick up the cubes.

B Challenges of Perspective Ray Casting

In Fig. 2, we present an additional motivation to adopt orthographic ray casting in MIRA.

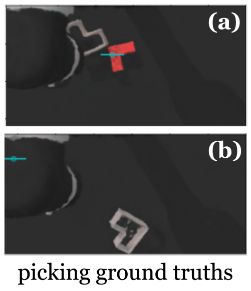

Figure 2: Challanges of Perspective Ray Casting. We visualize two images synthesized by NeRF using perspective ray casting. (a) A picking ground truth pixel that is proper for training. (b) An picking ground truth pixel that is occluded by the robot arm, which in turns fails the policy training. The occlusion happens because perspective ray casting needs to move the camera further away from the scene to perceive the whole scene. This adjustment accidentally captures the robot arm that occludes the ground truth pixel. In contrast, images rendered with orthographic ray casting does not depend on the distance between the camera and the scene. Therefore, one can easily tune the near / far plane to ignore the distractors (e.g., robot arm).

C Simulation Tasks

In the following table, we summarize the unique attributes of each simulation task in our experiments.

Task	precise placing	multi-modal placing	distractors	unseen colors	unseen objects
block-insertion	✓	✗	✗	✗	✗
place-red-in-green	✗	✓	✓	✗	✗
hanging-disks	✓	✗	✗	✓	✗
stacking-objects	✓	✗	✗	✗	✓

Table 2: Simulation tasks. We extend Ravens [1] with four new 6-DoF tasks and summarize their associated challenges.

D CUDA Kernel for Orthographic Ray Casting

```

inline __host__ __device__ Ray pixel_to_ray_orthographic(
  uint32_t spp,
  const Eigen::Vector2i& pixel,
  const Eigen::Vector2i& resolution,
  const Eigen::Vector2f& focal_length,
  const Eigen::Matrix<float, 3, 4>& camera_matrix,
  const Eigen::Vector2f& screen_center,
  float focus_z = 1.0f,
  float dof = 0.0f
) {
  auto uv = pixel.cast<float>().cwiseQuotient(resolution.cast<float>());

  Eigen::Vector3f dir = {
    0.0f,
    0.0f,
    1.0f
  };
  dir = camera_matrix.block<3, 3>(0, 0) * dir;

  Eigen::Vector3f offset_x = {
    (uv.x() - screen_center.x()) * (float)resolution.x() / focal_length.x(),
    0.0f,
    0.0f
  };
  offset_x = camera_matrix.block<3, 3>(0, 0) * offset_x;
  Eigen::Vector3f offset_y = {
    0.0f,
    (uv.y() - screen_center.y()) * (float)resolution.y() / focal_length.y(),
    0.0f
  };
  offset_y = camera_matrix.block<3, 3>(0, 0) * offset_y;

  Eigen::Vector3f origin = camera_matrix.col(3);
  origin = origin + offset_x + offset_y;

  return {origin, dir};
```
Listing 1: CUDA kernel for running orthographic ray casting in instant-NGP [2].

References

[1] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *CoRL*, 2020.

[2] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *SIGGRAPH*, 2022.