RL-LABEL: A Deep Reinforcement Learning Approach Intended for AR Label Placement in Dynamic Scenarios

Chen Zhu-Tian[1], Daniele Chiappalupi[1,2], Tica Lin[1], Yalong Yang[3], Johanna Beyer[1], Hanspeter Pfister[1]

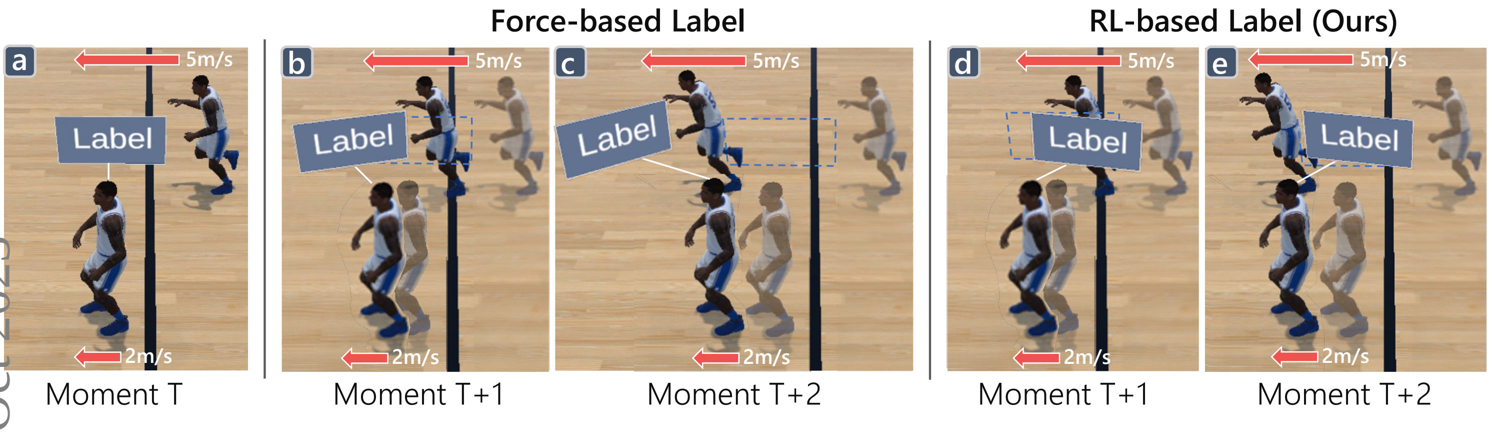


Fig. 1: RL-LABEL not only adapts label placement considering players' current motion status (e.g., speed, direction) and long-term outcomes but also ensures label stability over time. (a) Both players move left, with the rear player moving faster. A label is attached to the front player. (b)-(c) Using a force-based method, the label shifts left to avoid immediate occlusion but results in future occlusion. (d)-(e) With our method, the label moves right, sacrificing some immediate occlusion-free space for preventing future occlusion and ensuring stable visibility.

Abstract—Labels are widely used in augmented reality (AR) to display digital information. Ensuring the readability of AR labels requires placing them in an occlusion-free manner while keeping visual links legible, especially when multiple labels exist in the scene. Although existing optimization-based methods, such as force-based methods, are effective in managing AR labels in static scenarios, they often struggle in dynamic scenarios with constantly moving objects. This is due to their focus on generating layouts optimal for the current moment, neglecting future moments and leading to sub-optimal or unstable layouts over time. In this work, we present RL-LABEL, a deep reinforcement learning-based method intended for managing the placement of AR labels in scenarios involving moving objects. RL-LABEL considers both the current and predicted future states of objects and labels, such as positions and velocities, as well as the user's viewpoint, to make informed decisions about label placement. It balances the trade-offs between immediate and long-term objectives. We tested RL-LABEL in simulated AR scenarios on two real-world datasets, showing that it effectively learns the decision-making process for long-term optimization, outperforming two baselines (i.e., no view management and a force-based method) by minimizing label occlusions, line intersections, and label movement distance. Additionally, a user study involving 18 participants indicates that, within our simulated environment, RL-LABEL excels over the baselines in aiding users to identify, compare, and summarize data on labels in dynamic scenes.

Index Terms—Augmented Reality, Reinforcement Learning, Label Placement, Dynamic Scenarios

1 INTRODUCTION

Many augmented reality (AR) applications (e.g., sports analytics [25, 58], mechanical maintenance [32], and education [49,57]) use labels to display digital information. A label is a small virtual canvas connected to physical objects via leader lines. Determining label layouts is essential to help users perceive the visual information. Thus, extensive research has explored automatic generation of AR label layouts, with most methods treating it as an optimization problem [3]. These view management systems aim to maximize objectives such as preventing label-object overlaps, avoiding leader line intersections, and minimizing distances between labels and their physical referents [3]. Though these methods work well for static scenarios, they often fall short in dynamic scenarios where physical objects, like athletes in sports games, are constantly moving. This presents a clear need for improved AR label management systems that can adapt to changing object positions.

Managing label layouts for dynamic objects in AR presents unique challenges. Labels need to adapt their positions continually and stably to moving objects while avoiding occlusion and line intersections. Thus, view management systems should generate layouts that are optimal not only for the current moment but also capable of transitioning smoothly to future layouts. However, existing methods primarily focus on optimizing layouts for the present, often neglecting future scenarios. This leads to sub-optimal or unstable layouts over time. For instance, one most widely-used method for label management is the force-based algorithm [3], which models objectives as forces (e.g., occlusion-free placement is modeled as repulsive forces between labels and objects). Consider a scenario where both players in Fig. 1a are moving to the left, with the rear player moving faster and a label attached to the front player. A force-based method would push the label to the left to avoid occluding the rear player (Fig. 1b). However, this would result in severe future occlusion since the rear player also moves to the left (Fig. 1c).

We approach the AR labels placement problem for moving objects

- [1] *Harvard John A. Paulson School of Engineering and Applied Sciences*
- [2] *ETH Zurich, Zurich*
- [3] *Virginia Tech*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

from a new angle: the view management system should generate label layouts based on both current and predicted future states. Inspired by other applications in dynamic real-world environments (e.g., self-driving cars, robotics control), we achieve this goal by employing a reinforcement learning (RL) method. Specifically, we propose RL-LABEL, an RL-based view management system for scenes with multiple moving objects, each with an attached label. To the best of our knowledge, we are the first to explore using RL to manage AR labels for moving objects. RL-LABEL observes object and label states, such as positions and velocities, and the user's viewpoint, to make informed decisions about label placement that balance the trade-offs between immediate and long-term objectives. For instance, RL-LABEL moves the label to the right, while temporarily causing occlusion of the players (Fig. 1d), to prevent future occlusion (Fig. 1e).

To ensure reproducibility, we tested RL-LABEL within simulated AR environments based on Virtual Reality and two real-world trajectory datasets (one for NBA players and the other for students on a campus). Computational experiments demonstrate that RL-LABEL can effectively learn the decision process to achieve a long-run optimization, outperforming two baselines (i.e., no view management and a force-based method) by reducing occlusions, line intersections, and movement distance of the labels. Furthermore, we conducted a formal user study with 18 participants in the same simulated environments to compare RL-LABEL with the two baselines in assisting users in three visual search tasks, i.e., identify, compare, and summarize data on AR labels. Overall, within the simulated environments, users performed these tasks faster, more accurately, and with a lower mental load when using RL-LABEL than the baselines.

In summary, our contributions are threefold: First, we formulate the label placement in dynamic scenes as a sequential decision (instead of an optimization) problem with the goal to maximize cumulative rewards; Second, we design and develop RL-LABEL, an RL-based method to solve this problem; Third, we evaluate RL-LABEL with both computational experiments and user studies, showing that it outperforms force-based methods in both quantitative and qualitative aspects.

2 RELATED WORK

Labels in Augmented Reality. Labels have long been used to annotate objects in illustrations [3]. Since the early '80s, automated view management systems for label placement have emerged in computer graphics research [1], varying by label type (*internal* or *external*) and environment (*desktop* or *AR*). This work concentrates on placing external labels in AR environments.

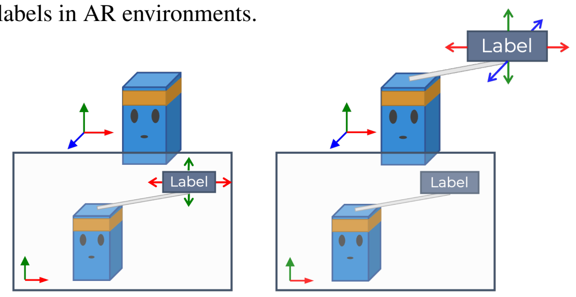


Fig. 2: Left: 2D labels are placed and managed in the image space. Right: 3D labels are placed and managed in the world space, and subsequently projected onto the image plane.

Early view management systems for AR environments mainly place labels in 2D screen space (Fig. 2 Left), unaffected by 3D transformations. Representative examples include Azuma and Furmanski's identification of label clusters in screen space [2], visual saliency driven label placement [14, 17, 38], and Tatzgern et al.'s implementation of adaptive clustering based on information density to reduce visual clutter [48]. Yet, 2D labels face the "floating labels" issue [47], where frequently changing user viewpoints lead to unpredictable label positions due to changing projected 3D points during camera movements.

Unlike 2D labels, 3D labels (Fig. 2 Right) offer greater stability in AR when users change viewpoints. Tatzgern et al. [47] propose Hedgehog labeling, a representative example of 3D labels. They use 3D geometric constraints to generate label placements that fulfill the desired objectives (e.g., occlusion-free) and are stable over time, even when the viewpoint changes. Madsen et al. [30] later compared this method to 2D labels in an empirical study. Findings revealed that 3D labels with a limited update rate outperform 2D and continuously updated 3D labels. More recently, Koppel et al. [22] contributed a system for 3D AR labels that manages label visibility and level of detail, considering both the labels in front of and out of the view. Gebhardt et al. [13] utilize RL to control the visibility (i.e., show or hide) of an object's label based on the user's gaze data. Lin et al. [26] studied the design space of 2D and 3D AR labels for out-of-view objects.

While existing methods have proven useful and effective in various AR scenarios, they primarily focus on static objects with fixed positions. However, in real-world environments, objects often move dynamically (e.g., basketball players, vehicles, conveyor belt sushi). This presents additional challenges for label placement, which should consider both the current and predicted future states of the objects. We aim to develop a view management system that observes dynamic moving objects and adapts 3D labels in real-time.

Adaptive User Interfaces in Augmented Reality. A label is a type of user interface (UI) that displays visual content in a small canvas [3]. Unlike traditional desktop UIs, many design decisions of AR UIs, such as display location and manner, cannot be predetermined and must adapt to the user's context in real-time [15]. Extensive research has focused on making AR UIs adaptive by leveraging geometry information from the environments. For example, AR UIs can be aligned with edges [34], placed on surfaces [56], and interact with 3D meshes [11, 12] extracted from the physical environments.

In addition to basic geometry information, recent AR UIs leverage semantic information of the scene to enhance the user experience. For instance, Tahara et al. [45] employ a scene graph to define the spatial relationships between virtual and physical objects, automatically adjusting the virtual content when the user moves to other environments. AdapTutAR [18], an AR task tutoring system, adjusts the teaching content adaptively based on the user's characteristics. SemanticAdapt [7] uses computer vision techniques to detect the category of real objects and associates them with AR content. Lindlbauer et al. [28] control the placement and level of detail of virtual content by considering both the indoor environment and the cognitive load of the user's performing task. ScalAR [37] enables designers to author semantically adaptive AR experiences in VR. However, all these works focus on static scenarios where the physical objects are placed in fixed positions. In contrast, we target dynamic scenarios where objects are moving.

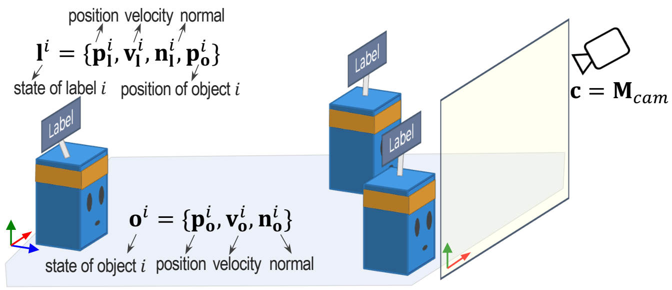
AR Labeling as a Partially Observable Decision Problem. Fundamentally, managing labels for moving objects in AR involves dealing with a partially observable decision problem [19]. In a partially observable environment, the entire state is partially visible to external sensors. For instance, an object's destination and planned route are known only to the object itself. This distinguishes AR from other 3D environments, where the system state is fully visible. Recently, deep RL has demonstrated promising performance in addressing partially observable problems in applications such as self-driving cars [21], robotics control [29], and video games [42]. Inspired by these applications, this work explores using deep RL to manage labels for moving objects.

Reinforcement Learning for Data Visualizations. Unlike supervised learning, which relies on historical data, RL models learn from interactive experiences with the environment (e.g., play GO), making RL suitable for dynamic scenarios requiring long-term optimization.

Only a few studies have explored RL methods for solving visualization problems. For instance, PlotThread [46] uses RL to create user-preferred storyline layouts. MobileVisFixer [51] employs RL methods to adapt visualizations for different mobile devices by adjusting parameters like size, offset, and margin. Table2Chart [54] uses deep Q-learning to recommend chart templates based on input data tables. In summary, most existing works focus on static 2D environments. We aim to apply RL-based methods to manage label layouts in dynamic 3D environments, an underexplored area with unique challenges.

3 Problem Formulation

This work, similar to Yao et al. [52], focuses on scenarios where the viewer is stationary but can rotate their viewpoint horizontally or vertically to observe multiple moving objects, such as watching sports in a stadium or monitoring environments using surveillance cameras. We constrain that each object is annotated by a single label and leave multiple-label or moving viewpoint scenarios for future research.



position velocity normal
$\mathbf{l}^i = \{\mathbf{p}_\mathbf{l}^i, \mathbf{v}_\mathbf{l}^i, \mathbf{n}_\mathbf{l}^i, \mathbf{p}_\mathbf{o}^i\}$
state of label i
position of object i
Label
Label
Label
$\mathbf{c} = \mathbf{M}_{cam}$
$\mathbf{o}^i = \{\mathbf{p}_\mathbf{o}^i, \mathbf{v}_\mathbf{o}^i, \mathbf{n}_\mathbf{o}^i\}$
state of object i
position velocity normal

Fig. 3: We consider scenarios where a stationary viewer observes multiple moving objects, each with a corresponding label.

At each time step, the view management process involves three components (Fig. 3):

- **Object:** An object $\mathbf{o}^i$ can be approximated as a cube shape rigid body in the 3D *world* space $\mathbb{R}^3_{wld}$. The object $\mathbf{o}^i$ attempts to move to a goal position following a planned route and a preferred velocity. For a view management system, the observable state of the object can be represented as $\mathbf{o}^i = \{\mathbf{p}_\mathbf{o}^i, \mathbf{v}_\mathbf{o}^i, \mathbf{n}_\mathbf{o}^i\}$, where $\mathbf{p}_\mathbf{o}^i = (x^i, y^i, z^i)$, $\mathbf{v}_\mathbf{o}^i = (v_x^i, v_y^i, v_z^i)$, and $\mathbf{n}_\mathbf{o}^i = (n_x^i, n_y^i, n_z^i)$ are its position, velocity, and normal, respectively. The observable state of all the objects is thus $\mathbf{o} = \{\mathbf{o}^i\}_{i=1}^n$, where n is the number of objects.
- **Label:** A label $\mathbf{l}^i$ is a rectangle shape canvas in the 3D *world* space $\mathbb{R}^3_{wld}$, linking to its target object $\mathbf{o}^i$ through a leader line. The observable state of the label can be defined as $\mathbf{l}^i = \{\mathbf{p}_\mathbf{l}^i, \mathbf{v}_\mathbf{l}^i, \mathbf{n}_\mathbf{l}^i, \mathbf{p}_\mathbf{o}^i\}$, where $\mathbf{p}_\mathbf{l}^i$, $\mathbf{v}_\mathbf{l}^i$, and $\mathbf{n}_\mathbf{l}^i$ are its position, velocity, and normal, respectively; $\mathbf{p}_\mathbf{o}^i$ represents the other endpoint of the leader line. In practice, the label usually keeps its normal $\mathbf{n}_\mathbf{l}^i$ pointing to the camera to maximize its readability. Similarly, the state of all labels is represented by $\mathbf{l} = \{\mathbf{l}^i\}_{i=1}^n$, where n is the number of labels, which equals the number of objects.
- **Viewpoint:** A viewpoint is determined by the camera $\mathbf{c}$ and specifies the observer's position in relation to the objects and labels being watched. Thus, the state of the viewpoint $\mathbf{c}$ can be represented as the projection matrix $\mathbf{M}_{cam}$.

For a label $\mathbf{l}^i$, a view management system should generate an action $\mathbf{a}^i$ to update its position based on its current state $\mathbf{l}^i$, the states of its target object $\mathbf{o}^i$, other labels $\mathbf{l} \setminus \{\mathbf{l}^i\}$, other objects $\mathbf{o} \setminus \{\mathbf{o}^i\}$, and the viewpoint $\mathbf{c}$. We simplify these states as $\mathbf{s}^i = \{\mathbf{l}^i, \mathbf{o}^i, \mathbf{l} \setminus \{\mathbf{l}^i\}, \mathbf{o} \setminus \{\mathbf{o}^i\}, \mathbf{c}\}$ and regard this process as a mapping function:

$$\pi(\mathbf{s}^i) \mapsto \mathbf{a}^i \tag{1}$$

This mapping process is applied on each label at each timestamp.

Typically, a view management system should output actions to optimize the objectives. We can define a reward function $r(\mathbf{s}^i, \mathbf{a}^i)$ to evaluate how good the action $\mathbf{a}^i$ achieves the objectives given the input $\mathbf{s}^i$. A good view management system for dynamic scenarios should maximize the reward in the long run. This can be formulated in an RL framework [44]:

$$\arg\max_{\pi} \{r(\mathbf{s}^i, \mathbf{a}^i) + \gamma V^*(\mathbf{s}^i, \mathbf{a}^i) | \pi(\mathbf{s}^i) \mapsto \mathbf{a}^i\} \tag{2}$$

where γ is a discount factor, V^* is the optimal value function that estimates the optimal cumulative rewards to the future after performing action $\mathbf{a}^i$ under state $\mathbf{s}^i$. Simply put, Equation 2 describes that the optimal view management system π should generate actions to maximize not only the reward $r(\mathbf{s}^i, \mathbf{a}^i)$ for the current state but also the future cumulative rewards $\gamma V^*(\mathbf{s}^i, \mathbf{a}^i)$.

Similar to previous works [5, 6], we aim to solve Eq. 2 using deep RL. Specifically, we train a neural network to approximate an optimal mapping function π. To assist in the training, we also train a neural network to approximate the optimal value function V^*.

The difference between RL- and optimization-based methods. Most of the existing systems for static objects formulate the view management problem as an optimization problem that aims to optimize the reward for the *current state* [3] (i.e., the first part of Eq. 2, $\arg\max_\pi\{r(\mathbf{s}^i, \mathbf{a}^i)\}$). They excel when the scene is static, as the current state will not change over time. However, in dynamic scenarios, where the state changes continuously, these methods tend to underperform because they cannot predict the *future states* of objects. Consequently, they may not provide the optimal or stable layout over time. In contrast, RL-based methods have the potential to overcome this issue by considering both current and future states.

The challenges of placing labels in AR vs. other 3D dynamic scenarios. Placing labels on moving objects in AR environments presents a unique challenge compared to other similar 3D dynamic scenarios, such as VR or 3D games. In 3D games, the system can access the full states of objects and thus can *plan* the best moving trajectories of the labels in advance. In contrast, in AR environments, the system can only observe the current state of the objects and is uncertain about their future states, such as their velocities. Therefore, the system must have the capability to predict future states to place the labels properly.

4 RL-Label Design

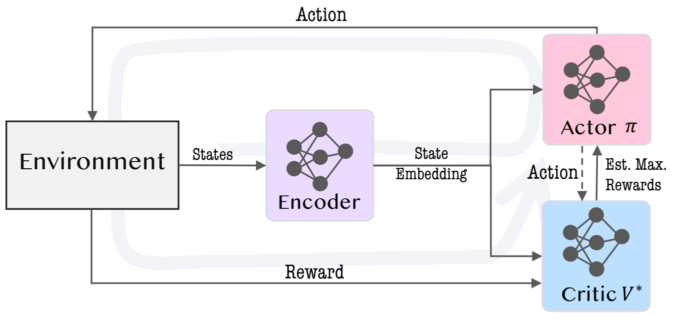


Fig. 4: Our RL-based method consists of three components: an *Encoder* that encodes the current state of the environment, an *Actor* that generates actions for placing the labels, and a *Critic* that evaluates the generated actions and provides feedback to improve the actor based on the rewards obtained from the environment.

We introduce RL-Label, which uses an *Actor-Critic* framework [44] to manage the label placements of moving objects. We chose the Actor-Critic framework since it has demonstrated state-of-the-art performance in various domains and benchmarks [10, 16, 31, 40]. It consists of three main components (Fig. 4):

1. **Encoder for State Embedding** (Sec.4.1). One of the main challenges of this work comes from state heterogeneity (i.e., associating each label with the viewpoint and objects) and variability (i.e., encoding the relationship between each label and its neighbors, which can vary in number). We address the challenge through space transformations and a neural network with a self-attention mechanism.
2. **Actor for Action Generation** (Sec.4.2). We then use an Actor network to generate actions to place labels based on the state embedding. The challenge lies in designing an appropriate action space that enables the model to converge during training and generate effective actions to achieve objectives. Our action space consists of two accelerations (x- and z-) that control the movement of a label in an x-z square plane on top of its target object.

3. **Critic for Reward Learning** (Sec.4.3). To improve the effectiveness of the actions generated by the Actor network, we introduce a Critic network that predicts the future cumulative reward for the actions based on the current environment states and rewards. The environmental rewards are designed to incorporate the objectives of the label layout, such as avoiding occlusion, line intersections, and jittering. The Critic network is solely utilized to aid in training the Actor network and is not used during inference.

4.1 Encoder for State Embedding

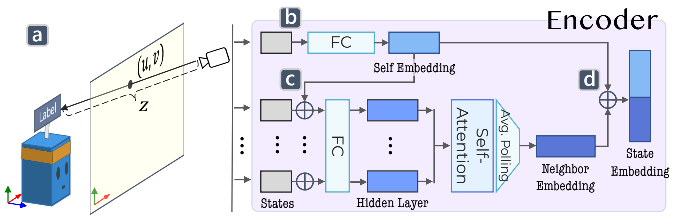


Fig. 5: For each label or object, the Encoder considers its state (e.g., position) relative to the camera and neighboring objects. It then employs a neural network to embed the label or object's state into a high-dimensional vector.

To train the neural network, it is necessary to convert the state information of labels, objects, and the viewpoint into a vector. We achieve this by a label-centered space transformation and a neural network with a self-attention mechanism.

Space Transformation. To place a label $\mathbf{l}^i$, the view management system needs to consider its state in relation to other labels, objects, and the viewpoint. We achieve this in two steps (Fig. 5a):

1. *Neighbor Encoding.* To encode the states of the label's neighbors (*i.e.*, other objects and labels), we transfer their positions and velocities into the label's local space. This captures their relative positions and movements with respect to the label. Additionally, to distinguish between objects and labels, we append a binary value w to each neighbor vector. A value of 1 indicates that a neighbor is an object, while a value of 0 indicates that the neighbor is a label.

2. *Viewpoint Encoding.* To ensure that the label does not occlude other objects and labels, it is necessary to associate the viewpoint's state with the objects and labels. To encode the viewpoint's state, we transfer all objects' and labels' positions to *ray space* $\mathbb{R}^3_{ray}$, in which a point $\mathbf{p}_{ray}$ is defined as $(u_{scn}, v_{scn}, z_{cam})$, where (u_{scn}, v_{scn}) is the point's position in screen space $\mathbb{R}^2_{scn}$ and the third coordinate z_{cam} specifies the distance from the camera to the point's 3D position in $\mathbb{R}^3_{cam}$. Such transformation can retain both the 2D and 3D information.

The resulting states after the two steps are referred to as *encoded states*. Specifically, the states of the label $\mathbf{l}^i_{ray}$, a neighbor object $\mathbf{o}^j_{ray}$, and a neighbor label $\mathbf{l}^j_{ray}$ are encoded as:

$$\mathbf{l}^i_{ray} = \{\mathbf{p}^i_{\mathbf{l},ray},\ \mathbf{v}^i_{\mathbf{l}},\ \mathbf{n}^i_{\mathbf{l}},\ \mathbf{p}^i_{\mathbf{o},ray} - \mathbf{p}^i_{\mathbf{l},ray}\}$$
$$\mathbf{o}^j_{ray} = \{\mathbf{p}^j_{\mathbf{o},ray} - \mathbf{p}^i_{\mathbf{l},ray},\ \mathbf{v}^j_{\mathbf{o}} - \mathbf{v}^i_{\mathbf{l}},\ \mathbf{n}^j_{\mathbf{o}},\ w:1\}$$
$$\mathbf{l}^j_{ray} = \{\mathbf{p}^j_{\mathbf{l},ray} - \mathbf{p}^i_{\mathbf{l},ray},\ \mathbf{v}^j_{\mathbf{l}} - \mathbf{v}^i_{\mathbf{l}},\ \mathbf{n}^j_{\mathbf{l}},\ \mathbf{p}^j_{\mathbf{o},ray} - \mathbf{p}^i_{\mathbf{l},ray},\ w:0\}$$

State Embedding. Following the space transformation, we utilize a neural network to embed the resulting states into a high-dimensional space. The network embeds the label $\mathbf{l}^i$ and its neighbors differently:

1. *Self Embedding* (Fig. 5b). We first concatenate the encoded state of the label $\mathbf{l}^i_{ray}$ with the encoded state of its target object $\mathbf{o}^i_{ray}$ and then embed them into a 128-length vector using a fully connected (FC) network.

2. *Neighbor Embedding* (Fig. 5c). Building on previous work [5], we embedded the neighbors of the label $\mathbf{l}^i$ as follows:

   For each of the label's neighbors (except for its target object), we first concatenate its encoded state (e.g., $\mathbf{l}^j_{ray}$ or $\mathbf{o}^j_{ray}$) with the 128-length embedding vector obtained from the previous step. Next, we embed the concatenated vector into a 128-length vector using an FC network. Since the number of neighbors can vary across different scenes, we use a *self-attention mechanism* [50] to address the variable number of inputs. This network mechanism learns the relative importance of each neighbor as attention scores and uses the scores as weights to combine all the inputs into a fixed-size output.

Finally, we concatenate the embedding of the label and its neighbors into a single vector (Fig. 5d), which contains rich information about the label, its neighbors, and the viewpoint of the current moment.

4.2 Actor for Action Generating

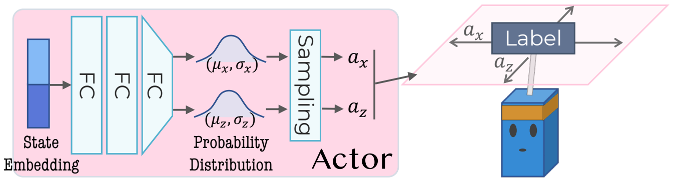


Fig. 6: An Actor network generates actions based on the state embedding to place the label. The label's movement is constrained to a two-dimensional x-z square plane on top of its target object, and the available actions consist of x- and z- accelerations for the label.

The system should generate actions to place the labels. To achieve this, we first define an action space that contains all possible actions, and then use an Actor network to learn a policy (i.e., π) that generates actions from this space.

Action Space. The action space plays a central role in successfully training a model: a large action space can make the model fail to converge while a small one may not be able to generate effective actions that fulfill the objectives (e.g., occlusion free, stable movements). Inspired by Tatzgern et al. [47], we first constrain the label's movement to an x-z square plane on top of its target object (Fig. 6, right). We choose this plane to ensure consistency in the degrees of freedom of the labels and objects, as we assume that the objects can only move in the x and z dimensions. We then define the actions generated by the model as x- and z- accelerations (i.e., a_x and a_z) that control the label's movement to ensure stability. In other words, the action space consists of two accelerations that control the label to move in an x-z square plane on top of its target object. In our study, we also explored another action space in which the model generated the x and z velocities or positions of the labels, but this led to jittering movements due to the non-deterministic nature of the network output (see below).

Action Generation. We use a three-layer FC network to learn a mapping function π that generates actions. The network inputs the embedding vector of label $\mathbf{l}^i$ and outputs the means and standard deviations of two normal distributions. These two distributions serve as probability distributions to sample the accelerations for the x and z directions of the label (Fig. 6, left). This stochastic policy encourages the network to explore different actions during training, rather than only selecting the same action every time, which can help reduce the possibility of getting stuck in a locally optimal solution and has been widely used in RL to improve performance [44].

4.3 Critic for Reward Learning

To improve the Actor network's ability in generating effective actions, we designed rewards to reflect the feedback from the environment, incorporating the objectives of the label layout, such as avoiding occlusion, line intersections, and jittering. However, these rewards only

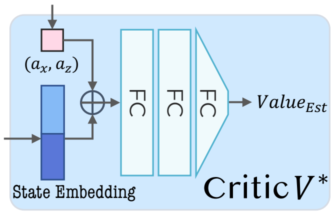


Fig. 7: The Critic network estimates the future accumulated reward for the actions generated by the Actor, based on the current state of the environment. The estimated reward is then used to refine the Actor.

provide feedback for the current moment. To improve long-term performance, we use a Critic network (V^*). The Critic learns from the rewards, actions, and states to predict the future cumulative reward of a given action on the current state, guiding the Actor network.

Reward Design. In line with previous work [3], we aim for the labels to move without occlusion, line intersection, and jittering. Thus, we design the reward from the environment as $r(\mathbf{s}^i, \mathbf{a}^i) = r_{occ} + r_{int} + r_{a_{x,y}}$, which awards achieving each objective while penalizing any failures. Specifically, each term is defined as follows:

$$r_{occ} = \begin{cases} -0.1 n_{occ}, & \text{if } n_{occ} > 0 \\ 0.1, & \text{otherwise} \end{cases}$$

$$r_{int} = \begin{cases} -0.1 n_{int}, & \text{if } n_{int} > 0 \\ 0.1, & \text{otherwise} \end{cases}$$

$$r_{a_{x,y}} = \begin{cases} 0.001, & \text{if } |a_x| \leq \max_{a_x} \text{ and } |a_y| \leq \max_{a_y} \\ -0.001, & \text{otherwise} \end{cases}$$

where n_{occ} represents the number of objects occluded by a label, n_{int} represents the number of lines intersected by the label's leader line, and $\max_{a_x}$ and $\max_{a_z}$ represent the predefined maximum accelerations on x- and z- dimensions. We chose the number of rewards (e.g., 0.1, 0.001) based on prior works [5, 20] and empirical evaluations.

Reward engineering [9] is an iterative process, similar to fine-tuning hyperparameters. We arrived at our current design through extensive exploration of reward designs, including variations in positive/negative reward values, reward magnitudes, incorporation of occlusion area, and measuring label movement distances to assess jittering. None of them demonstrated the same level of performance as our current design.

Cumulative Rewards Prediction. To help the Actor network generate actions that maximize both the reward for the current state and the future cumulative rewards, we used three FC layers to approximate the optimal value function V^*. This network predicts the future cumulative reward based on the current state and the action generated by the Actor. The predicted reward is then used to calculate the loss to train the Actor. Note that the Critic network is only used as an assistant during training and is not necessary for inference [44].

5 Dataset and Network Training

To train and evaluate RL-Label, we collected two human movement datasets to simulate real-world scenarios with moving objects (Sec. 5.1). We first introduce the training process (Sec. 5.2) and then report the quantitative evaluation results (Sec. 6).

5.1 Dynamic Environments Simulation

Real-world Human Movement Datasets. We used two popular human movement datasets to simulate real-world environments with moving objects. The first dataset, *NBA* [43], consists of the trajectories of 10 players on the court during the first quarter (720 seconds in total) of a well-known NBA game [1]. The second dataset, *STU* [23], records the trajectories of students in an open campus environment over a period of 400 seconds. Unlike the NBA dataset, the number of objects over time in the STU dataset is dynamic, as some students enter or leave the campus environment during the period. Both datasets represent the trajectory of an object (i.e., player or student) as a list of 2D positions taken at an interval of 0.04 seconds. In total, the NBA and STU datasets contain over 180K and 160K positions, respectively.

Table 1: Statistics of scenes in the two datasets.

Dataset	Avg. Max. #Objects	Avg. Speed (m/s)	Avg. Moving Distance (m)
NBA	10	1.88	28.19
STU	20	1.29	9.82

We preprocessed the NBA and STU datasets by dividing them into small scenes lasting 15 seconds each, so that there is no overlap between objects' trajectories in different scenes. We excluded scenes that are less than 15 seconds long. For the NBA dataset, we further removed scenes where the game stops due to events such as fouls or substitutions. Ultimately, we gathered 26 scenes for each dataset. Table. 1 summarizes the statistics of the scenes for each dataset. The STU dataset is generally more challenging due to the larger and dynamic number of objects.

Simulated Environments in Unity. We used Unity to simulate the scenes in the two datasets. Specifically, we created 3D cubes to represent the objects in the scenes and updated the cubes' positions every 0.04 seconds to follow the objects' trajectories. An example scene for each dataset is shown in Fig. 8a and b.

5.2 Network Training

We followed machine learning conventions and randomly split the scenes of each dataset into two sets: 80% of the scenes for training and 20% of the scenes for testing. This resulted in 20 scenes for training and 6 scenes for testing in each dataset.

Loss Function. Our method adopts an actor-critic framework [44], which usually involves two loss terms:

[1]Golden State Warriors and Cleveland Cavaliers on Dec 25th, 2015

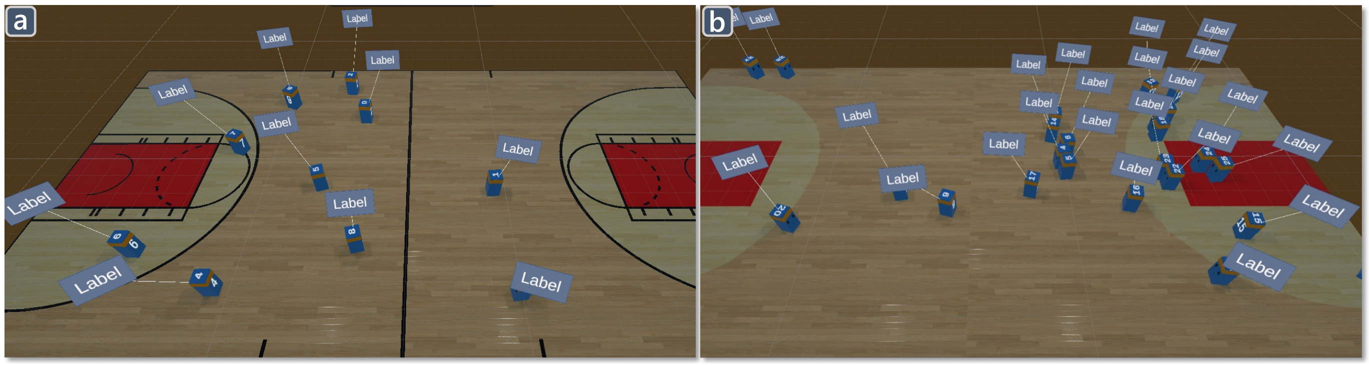


Fig. 8: a) the NBA dataset, in which each scene contains 10 fast-moving objects. b) the STU dataset, in which the number of objects is dynamic. Objects in the STU dataset move slower than those in the NBA dataset.

$$\mathcal{L}_\pi = -\frac{1}{nT}\sum_{i=0}^{n}\sum_{t=0}^{T}\log\pi(\mathbf{a}^{i,t}|\mathbf{s}^{i,t})A_{i,t} \quad (3)$$

$$\mathcal{L}_V = -\frac{1}{nT}\sum_{i=0}^{n}\sum_{t=0}^{T}(V^*(\mathbf{s}^{i,t},\mathbf{a}^{i,t}) - R_t)^2 \quad (4)$$

where T represents the maximum time step of a scene, n is the number of labels. Eq. 3 penalizes the actor's actions that have a high probability of occurring but result in low advantages $A_{i,t} = r(\mathbf{s}^{i,t},\mathbf{a}^{i,t}) + \gamma V^*(\mathbf{s}^{i,t},\mathbf{a}^{i,t}) - V^*(\mathbf{s}^{i,t-1},\mathbf{a}^{i,t-1})$, which is an estimation of the future cumulative rewards with bias subtraction for the actions in the current state. Eq. 4 trains the critic's estimation to match the future cumulative rewards, $R_t = \sum_{t'=t}^{T}\gamma^{t'-t}r(\mathbf{s}^{i,t'},\mathbf{a}^{i,t'})$. In our implementation, we use PPO [40], a state-of-the-art actor-critic algorithm, which involves several advanced techniques to calculate Eq. 3, such as using generalized advantage estimation [39] to compute $A_{i,t}$, differential entropy loss to encourage exploration and avoid premature convergence, and clipped surrogate objective to stabilize the training process. We refer the readers to Schulman et al. [40] for more technical details.

Hyper Parameters. Two hyper parameters play important roles in our method, namely, *maxAcc* and *numAgent*:

- *maxAcc* controls the absolute value of the maximum acceleration the actor can apply for the labels. A larger *maxAcc* can reduce occlusions but increases the jittering of the labels. Based on the mean moving speed of objects, we set the *maxAcc* for the NBA and STU datasets to $3m/s^2$ and $2m/s^2$, respectively.
- *numAgent* specifies the number of labels to be controlled by the model. Ideally, the model should control all the labels in a scene. However, a larger *numAgent* can make training more difficult. To address this issue, we employ a technique called Curriculum Learning [33], in which we gradually increase the *numAgent* during training. We increase the *numAgent* from 2 to 10 (stepsize: 2) and 4 to 20 (stepsize: 4) for the NBA and STU datasets, respectively.

Implementations Details. We use ML-Agent [20], a Unity-based reinforcement learning toolkit, to implement our system. ML-Agent provides an implementation of the RL networks using PyTorch [36], as well as components to connect the Unity environments with the PyTorch backends and manage the training. To accelerate the training, we also leverage the concurrent Unity instances feature provided by ML-Agent. Accordingly, we choose 204,800 and 1,024 for the buffer_size and batch_size to ensure the models collect rich enough experiences from the simulated environments. The learning rate is initialized as 3e-4 and decays linearly over training. Each training process contains 20 million steps, leading to about 1.3 million episodes (each episode lasts 150 steps). The training process for the models was conducted on servers equipped with NVIDIA V100 graphics cards and 6 vCPUs, and took approximately 5 hours to complete.

6 Computational Experiments

To evaluate our method, we first examined the effectiveness of the training process (Sec. 6.1), then qualitatively inspected the ability of the value network to predict future rewards (Sec. 6.2), compared our method with two baselines to assess its performance in handling occlusions, line intersections, and label jittering (Sec. 6.1), and finally discuss some typical behaviors of the model (Sec. 6.4).

6.1 Rewards Over the Training

Figure 9 demonstrates the rewards obtained by the RL model in the training (blue) and testing (purple) scenes during the training process. The plot indicates that both the rewards increase over the training process, indicating that the model is effectively learning and improving. The negative rewards in our approach stem from the strict reward definition, where occlusion is considered to occur whenever two labels overlap. However, from a human perspective, these occlusions can often be overlooked. Despite the presence of negative rewards, the scenes may still appear occlusion-free to the audience.

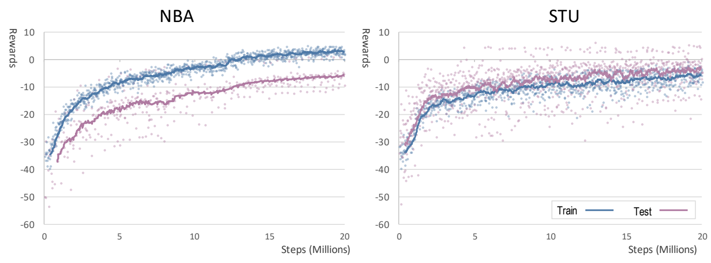


Fig. 9: Accumulated rewards per epoch in training (blue) and testing (purple) scenes for NBA (left) and STU (right) datasets.

6.2 Value Heatmap Learned by the Model

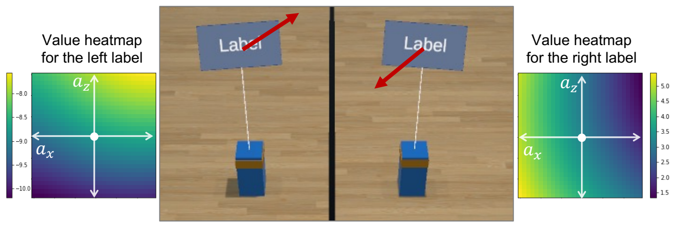

Value heatmap for the left label
Value heatmap for the right label
Label
Label
a_z
a_x

Fig. 10: Two value heatmaps generated by our model in a scenario where two objects move towards each other. Each heatmap represents the estimated future rewards of actions in the corresponding object's action space. Brighter colors indicate areas where the network considers a higher reward for moving, while darker colors indicate areas where the network does not prefer movement.

We conducted a qualitative inspection of the value network to assess its ability to estimate the accumulated rewards. We set up a scenario where two objects move towards each other and divided the action space into a 30x30 grid. Then, we estimated the rewards for different grids across the action space using the value network and visualized them as value heatmaps. Figure 10 displays examples of these heatmaps. The network suggests that the left label can achieve the maximum reward if it moves to the top right, while the right label should move to the bottom left. As a result, both objects avoid occluding each other. These examples demonstrate that the value network has learned to meaningfully estimate the consequences of different actions.

6.3 Performance in Achieving Objectives

We compared three different view management methods: 1) **No** view management as the baseline, where a label is fixed on top of its target object, 2) a **Force**-based method, and 3) **Ours**, the RL-based method. Since there is no existing view management system for labels of moving objects, we chose to implement and adapt Force based on Plane-based Hedgehog Labeling [47], a state-of-the-art method for managing 3D labels for static objects, under the guidance of their authors. Note that our comparison only intends to use Force as a reference instead of showing our method is better than Hedgehog Labeling [47].

Ideally, the view management system should reduce occlusions, line intersections, and the jittering of labels. We measure three metrics for each label to assess these three objectives:

- *OCC* measures the average number of objects and labels occluded by a label in each time step. A smaller OCC value is better.
- *INT* measures how many leader lines are intersected by a label's leader line on average in each time step, with a lower score indicating better performance.
- *DIST* measures the average extra moving distance of each label compared to its target object. DIST approximates the degree of jittering of the labels, and a smaller value is better.

Table 3 presents the results of the three methods on the three metrics. For OCC, without using a view management system, each label occludes 0.15 and 0.18 label and objects per time step in the NBA or STU

Table 2: Performance of three methods in reducing occlusion, line intersections, and extra movement distance of the labels.

	NBA			STU		
	OCC	INT	DIST	OCC	INT	DIST
No	0.15	0	0	0.18	0	0
Force	0.05	0.04	+11.29	0.07	0.06	+11.46
Ours	**0.04**	**0.02**	**+7.34**	**0.06**	**0.02**	**+2.58**

datasets, respectively. Both OURS and FORCE succeeded in reducing label occlusions by over two-thirds compared to the NO baseline. As for INT, without view management, it is zero since all the leader lines are perpendicular to the ground. Table 3 shows that the intersection introduced by OURS is half that of FORCE. Similarly, DIST should be zero without view management since the moving distance of a label should be the same as its target object. Table 3 indicates that OURS has much smaller DIST than FORCE, with OURS being almost one-fifth of FORCE in crowded STU scenarios. In general, these results show that OURS can reduce occlusions with fewer line intersections and more stable movements than FORCE.

6.4 Examples

We manually observed the model's control of the labels in all scenes to understand how it achieves the objectives. Here, we discuss some typical behaviors of the model. Video examples are also provided in the supplemental materials to illustrate these behaviors.

Avoiding occlusions through minimal movement. Figure 11a shows an example of the model's capacity to avoid occlusion by moving labels minimally. In this example, two objects are moving forward, with the rear object moving faster. To prevent occlusion of the rear object's label, the model controls the front object's label to move backward, using the small gap between the rear object and its label, rather than taking an easier but longer distance route (as indicated by the dashed arrow). This demonstrates the model's high controllability, with the task being completed within 0.2 seconds.

Avoiding line interactions through collaborative label movement. Figure 11b illustrates how the model controls the labels to move collaboratively to avoid line intersections in a complex movement scenario. The labels on the outside move in a circular path around the objects, similar to driving around a roundabout, while the label of the center object moves straight forward to the right. This strategy effectively eliminates any line intersections. This example demonstrates the model's ability to observe the states of a label's neighbors and adjust the label movement based on their relationships, highlighting its capability to handle complex situations.

Reducing Occlusion through strategic label placements. In Fig. 11c, the model is shown strategically placing the labels outside the court to reduce the likelihood of occlusion. When the objects move to the left side of the court, the model moves the labels to the empty space on the left, and vice versa when the objects are on the right. This behavior suggests that the model has learned to associate the empty space outside the court with safety based on the movement of objects and environmental feedback.

7 USER STUDY

We conducted a controlled user study to evaluate if RL-LABEL can help users perform visual tasks in dynamic scenes.

7.1 Experiment Settings

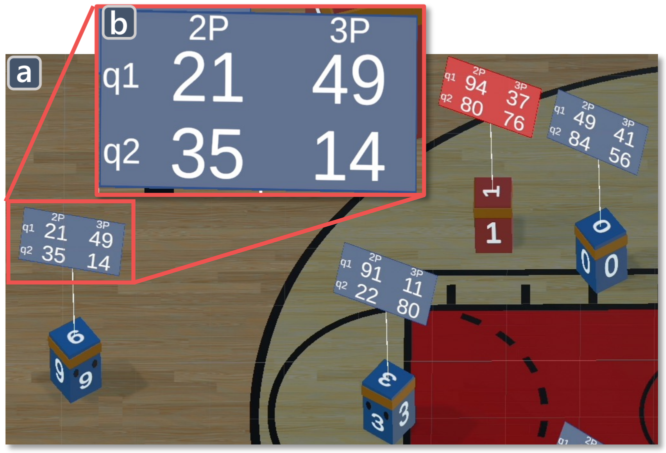


Fig. 12: a) A scene example of a task in which the objects are divided into blue and red teams. b) A table on a label, where the rows and columns represent quarters and points, respectively.

Participants and Apparatus. We recruited 18 participants (P1-P18; M = 8, F = 8; Age: 20 - 40) through university mailing lists and forums. Nine had no experience with AR/VR devices, seven had less than 1 year of experience, and one had 1-3 years of experience. Only one participant had no experience with data visualizations, while others had 0 to 5+ years of experience. The study was conducted in a quiet lab room using a standalone wireless Oculus Quest 2 VR headset, allowing participants to move freely in space without being limited by headset cables. The study took approximately 60 minutes, and each participant received a $20 gift card as compensation.

Conditions. We compared the three view management methods mentioned in Sec.6.3, i.e., NO, FORCE, and OURS.

Datasets. We reused the NBA and STU datasets for the user study. We used the scenes in the training set for practice trials and the scenes in the testing set for the actual study. In each scene, the objects are divided into blue and red teams. Each scene contains objects with a unique ID displayed on their bodies (Fig. 12a), divided into blue and red teams. Each object has a label attached to it, showing a static data

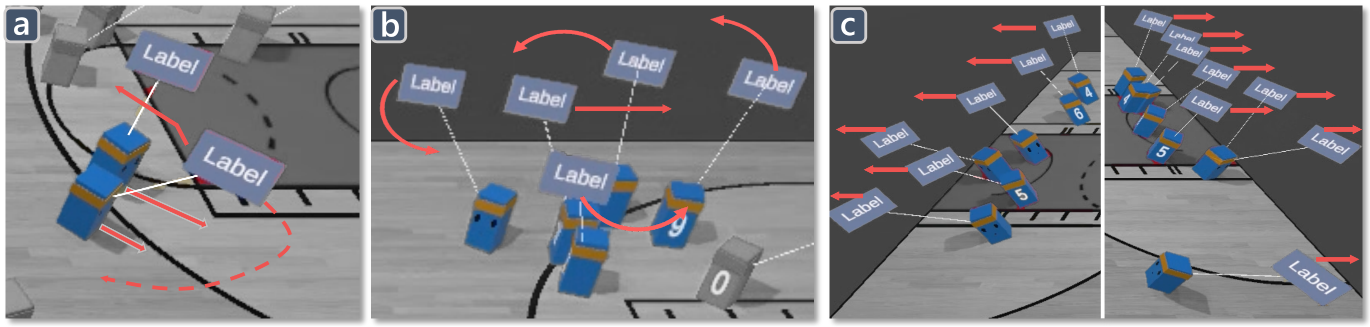


Fig. 11: Examples of the model's capabilities in controlling labels to avoid occlusion and line intersections. a) The model minimizes label movement by utilizing the small gap between an object and its label to avoid occlusion. b) The model controls labels to move collaboratively in a circular path around objects, eliminating any line intersections c) The model strategically places labels outside the court to reduce occlusion.

table (Fig. 12b), in which the rows and columns represent quarters and points, respectively. Each scene has a duration of 15 seconds.

Tasks. Following the previous work [26] on AR labels, we designed three basic visual search tasks [4]:

- `Identify`: This task asks the participants to identify the points a specific player scored in a specific quarter. The participants need to answer a question in the format of *"How many* <u>*[2, 3]*</u> *points has player* <u>*[X]*</u> *got in the quarter* <u>*[1, 2]*</u>*?"*, in which the underlined numbers are randomized across participants.
- `Compare`: This task asks the participants to compare the points the players in a specific team got in a specific quarter. The participants need to answer a question in the format of *"In the* <u>*[blue, red]*</u> *team, who got the most* <u>*[2, 3]*</u> *points in the quarter* <u>*[1, 2]*</u>*?"*, in which the underlined numbers are randomized across participants.
- `Summarize`: This task asks the participants to summarize the points a team scored in a specific quarter. The participants need to answer a question in the format of *"Overall, which team got more* <u>*[2, 3]*</u> *points in the quarter* <u>*[1, 2]*</u>*?"*, in which the underlined numbers are randomized across participants.

Study Design. We used the following full-factorial within-subject study design with Latin square-randomized order of the techniques:

	18	participants
×	3	techniques: No, Force, Ours
×	2	datasets: NBA, STU
×	3	tasks: `Identify`, `Compare`, `Summarize`
×	2	timed repetition
	648	total trials (36 per participant)

Participants were split evenly between the 3 Latin square-randomized technique orders. The orders of the datasets and tasks were kept consistent across all participants. Each task had randomized data on the labels and was repeated in different scenes.

Procedure. For each study, we first introduced the motivation, tasks, and general procedure, followed by obtaining consent (10mins). Before the actual tasks, participants completed a training session (10mins) to become familiar with the tasks, user interfaces, and to adjust the device until they were comfortable. The training session consisted of 18 = 3 (techniques) x 3 (tasks) x 2 (datasets) trials. Once the participant was confident, the experimenter proceeded to the actual tasks, which consisted of two sessions for the NBA and STU datasets, respecitvely. Each session (10mins) included 18 = 3 (techniques) x 3 (tasks) x 2 (datasets) x 2 (repetition) trials. For each trial, the participants were instructed to finish the task *as fast and accurately as possible* and to click a button to stop the timer once they were confident to answer. The scene in each trial looped until the participants clicked the button, after which it disappeared. After speaking their answer aloud, participants were asked to rate their mental load for the task. Participants were allowed to take a break between each session. At the end of the study, participants provided subjective feedback on each technique (5mins).

Measures. For each trial, we recorded the completion time (in seconds, from start to button click), accuracy (true or false), and the user's perceived mental load (on a 1-7 Likert scale).

7.2 Quantitative Results

We analyzed the statistical differences between the three methods in accuracy, completion time, and mental load. Overall, both Ours and Force are significantly better than No in all three measures and Ours slightly better than Force in completion time and mental load.

Table 3: Accuracy per method per task on the two datasets.

	NBA			STU		
	`Identify`	`Compare`	`Summarize`	`Identify`	`Compare`	`Summarize`
No	**100%**	83%	97%	50%	17%	**100%**
Force	97%	94%	**100%**	92%	67%	97%
Ours	97%	**97%**	97%	**97%**	**75%**	**100%**

Accuracy. Table 3 shows the accuracy of participants in different tasks using the three methods. The mean accuracy, in percentage, for NBA were: No = 93% (σ = 17%), Force = 97% (σ = 12%), and Ours = 97% (σ = 12%). For STU, the mean accuracy were: No = 56% (σ = 44%), Force = 85% (σ = 27%), and Ours = 91% (σ = 22%). A Shapiro-Wilk test reveals that the accuracy did not follow a normal distribution. Thus, we used a Friedman test with a null hypothesis that participants performed equally correctly with each method, which showed significant differences in the `Identify` (p = .0001) and `Compare` ($p < 0.0001$) tasks on the STU dataset. By further performing a Nemenyi post-hoc test, we found that participants performed significantly better in the `Identify` tasks with Ours (p = .0099) or Force (p = .0265) than with No. Similarly, participants performed significantly better in the `Compare` task with Ours (p = .001) or Force (p = .0018) than with No.

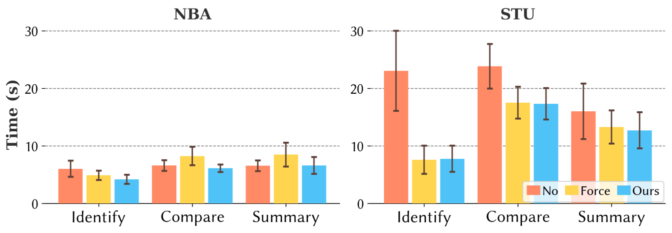


Fig. 13: Completion time per method per task on the two datasets.

Completion Time. Figure 13 presents the completion time of participants in different tasks using different methods with 95% confidence intervals (CIs). The respective mean times in seconds are No = 6.38 (σ = 3.25), Force = 7.20 (σ = 4.94), and Ours = 5.63 (σ = 3.20) on NBA, and No = 20.97 (σ = 16.15), Force = 12.79 (σ = 8.91), and Ours = 12.60 (σ = 8.95) on STU. Through a Shapiro-Wilk test, we confirmed that the completion times in each condition did not follow a normal distribution. Using a Friedman test with a null hypothesis that participants performed equally fast with each method, we found significant differences in the `Compare` (p = .0387) task on NBA, and in both the `Identify` ($p < 0.0001$) and `Compare` (p = .0293) tasks on STU. A Nemenyi post-hoc test revealed that participants completed the `Compare` task significantly faster with Ours than Force (p = .0355) on NBA, and significantly faster with Ours than No in the `Identify` (p = .001) and `Compare` (p = .0497) tasks on STU. Participants also completed the `Identify` (p = .001) and `Compare` (p = .0483) tasks significantly faster with Force than No on STU.

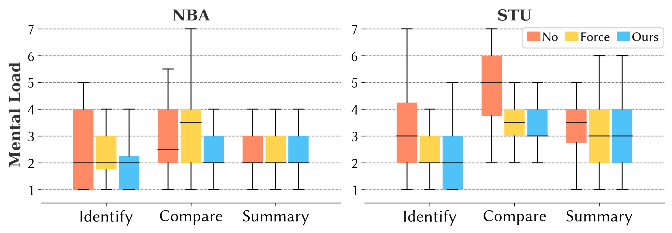


Fig. 14: Mental load per method per task on the two datasets.

Mental Load. Figure 14 shows the participants' subjective ratings on the mental load of using different methods in different tasks after each trial, where 1 means low and 7 means high mental load. The respective mean ratings are No = 2.55 (σ = 1.1), Force = 2.74 (σ = 1.29), and Ours = 2.36 (σ = 1.08) on NBA, and No = 3.83 (σ = 1.59), Force = 3.02 (σ = 1.27), and Ours = 3.01 (σ = 1.20) on STU. Since a Shapiro-Wilk test shown that the ratings did not follow a normal distribution, we used a Friedman test with a null hypothesis that the participants had the same mental load with the three methods. Findings revealed significant differences in the `Compare` (p = .0071) task on NBA and in both the `Identify` (p = .0004) and `Compare` ($p < .0001$) tasks on STU. After performing a Nemenyi post-hoc test, we found that participants reported significantly higher mental load with FORCE than with OURS in the `Compare` (p = .0483) task on NBA, and significantly higher mental load with NO than with OURS in both the `Identify` (p = .013) and `Compare` (p = .009) tasks on STU. Furthermore, participants reported significantly higher mental load with NO than with FORCE in both the `Identify` (p = .0075) and `Compare` (p = .001) tasks on STU.

7.3 Qualitative Feedback

We also collected qualitative feedback from the participants after the study. Participants were asked to comment on the pros and cons of each method. Overall, participants thought that OURS is the most promising method as it achieves occlusion-free and stable labels.

- NO. Participants provided mixed feedback on NO, which was primarily recognized for its label stability. Many participants appreciated the limited movement of the labels, mentioning that it made them "*easier to catch*" and that the "*length and direction did not change too much.*" However, a significant drawback noted by several participants was the excessive label occlusion, which made it difficult to read the labels when players were behind one another. One participant stated that the labels "*overlapped a lot, sometimes [making it] impossible to recognize the numbers.*"
- FORCE. By contrast, participants praised FORCE for its ability to eliminate label occlusion but complained about the excessive jittering and abrupt movement of the labels. Several participants noted the absence of overlapping labels, with one stating "*almost no occlusion at all.*" However, some participants found the unstable and unpredictable label movement distracting and "*hard to follow,*" or even "*made me dizzy.*" Although we believe that improved results can be achieved through an enhanced force-based method, it may still be prone to unpredictable movement, as it does not take into account the future states of the labels.
- OURS. Participants appreciated OURS primarily for its occlusion-free presentation and stable movement of labels. Many participants praised the method's ability to avoid label overlap effectively, as one user stated, "*Overlap is avoided well.*" The stable movement was also well-received, as users found it easy to track labels, e.g., "*movement of the labels is natural and easy to follow.*" However, some participants noted drawbacks, such as labels occasionally moving out of the screen and the layouts are "*not so appealing visually.*" Future improvements can involve optimizing the visual aesthetics of the labels as one objective or considering users' gaze and attention to enhance the local layout of the labels.

7.4 Summary

The user study demonstrated that resolving occlusions is crucial for reading labels in dynamic scenarios. OURS, the proposed method, was shown to effectively achieve this goal in both scenarios with fast-moving or many labels, and assist users with various visual search tasks, such as identifying, comparing, and summarizing data on moving labels. In addition, participants performed significantly better with OURS than with FORCE in comparing labels in the NBA dataset, suggesting that OURS can help users read multiple fast-moving labels. We attribute this to OURS's ability to stabilize label movements while resolving occlusions. Participants' subjective feedback also suggested that OURS can achieve both occlusion-free and stable movements of the labels.

8 DISCUSSION, FUTURE WORK, AND LIMITATIONS

AR Visualizations as Robots Without A Physical Body. Our RL method considers AR visualizations as robots without a physical body, which offers a key insight: if we can control robots to react and adapt to real-world environments, then we can do the same, or more, for AR visualizations. Although this concept is still in its early stages, it presents exciting opportunities for merging research in robotics with data visualization. By combining these two fields, we can create dynamic, interactive AR visualizations that adapt to real-world environments and assist users in making in-situ decisions. As we embark on this journey, we are excited to explore the potential of this innovative approach.

Involving Human Feedback for RL-based Visualizations. The very purpose of visualizations is to aid humans in gaining insights into the data presented. Therefore, when developing RL methods for visualizations, it is essential to incorporate human feedback in the training process. By doing so, we ensure that the AR visualization not only fulfills the optimization objectives but also meets the user's requirements and expectations that are often difficult to quantify. Recent studies in large language models have highlighted the importance of human feedback in RL [35]. Although we did not incorporate human feedback in the training process of our RL model as the first step, we are enthusiastic about exploring this direction in the future. Doing so would enable more advanced AR visualizations by, for example, incorporating aesthetics and level of detail in presenting the information.

Generalizing to More Complex and Diverse Scenarios. Our experiments on the STU datasets have already demonstrated the generalizability of our method to scenarios where the number of objects in the scene is continuously changing. Our method's data-driven nature enables us to apply it to other dynamic scenarios where objects or labels have different sizes, shapes, or anchor points as long as we incorporate those factors into the observable states and train the network with relevant data. In the future, we aim to expand our method to more complex and varied scenarios, such as city spaces with pedestrians and cars, similar to controlling mobile robots [5]. Furthermore, we are interested in exploring diverse action spaces that involve modifying the size, position (in the xyz coordinates), and opacity of the labels. Yet, this requires collecting additional movement datasets from the real world, which is beyond the scope of our current work. Nonetheless, our research lays the foundation for developing AR visualizations that can adapt to dynamic environments.

Extending to Real AR scenarios. In this study, we utilized VR to simulate AR environments, allowing the view management system to acquire accurate object states, such as positions and velocities. In real AR environments, such states can be obtained through the use of sensors like LiDAR [41]. However, these advanced sensors can be costly. A more cost-effective alternative is to use vision-based signals such as videos as input. For instance, Zhu et al. [55] developed an RL-based approach for navigating robots to locate a specified target using solely visual inputs. However, challenges such as real-time computing must be addressed in future research.

Study Limitations. While the model experiment and user study have demonstrated that RL-LABEL advances baselines, several limitations exist in our study. Firstly, we simplified the tasks by using cube shapes to represent humans and fixing the label size and viewer position. However, we believe that this is a necessary first step toward developing a more complex RL-based method, which is known to be more challenging to train compared to other machine learning models [24]. Secondly, we only evaluated users' performance on three fundamental visual tasks, while in-situ analytics with AR visualization can present more complex challenges. Finally, even though we followed exciting works [8, 26, 27, 53], we acknowledge that simulating AR in VR may not fully represent real-world scenarios. These limitations suggest potential areas for future research and improvements to RL-LABEL.

9 CONCLUSION

We introduced RL-LABEL, a novel RL-based method to address the challenges of managing AR label placements for moving objects. To the best of our knowledge, this is the first study to utilize RL for managing AR labels of moving objects. RL-LABEL takes into account both current and future predicted environment states to make optimal decisions regarding label placements. Our experimental results on two real-world trajectory datasets demonstrated that RL-LABEL effectively learned the decision-making process and outperformed two baselines in reducing occlusions, line intersections, and movement distance of the labels. The user study further showed that RL-LABEL improved user performance in various visual search tasks and achieved both occlusion-free and stable movements of the labels. Overall, our work establishes a strong foundation for the development of more advanced and effective RL-based methods for AR visualizations.

Acknowledgments

This research is supported in part by NSF award III-2107328, NSF award IIS-1901030, NIH award R01HD104969, and the Harvard Physical Sciences and Engineering Accelerator Award.

References

[1] J. Ahn and H. Freeman. A Program for Automatic Name Placement. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 21(2-3):101–109, 1984. doi: 10.1142/S0218001487000096

[2] R. T. Azuma and C. Furmanski. Evaluating Label Placement for Augmented Reality View Management. In *Proc. of ISMAR*, pp. 66–75. IEEE Computer Society, Los Alamitos, 2003. doi: 10.1109/ISMAR.2003.1240689

[3] M. A. Bekos, B. Niedermann, and M. Nöllenburg. External Labeling Techniques: A Taxonomy and Survey. *Comput. Graph. Forum*, 38(3):833–860, 2019. doi: 10.1111/cgf.13729

[4] M. Brehmer and T. Munzner. A Multi-Level Typology of Abstract Visualization Tasks. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2376–2385, 2013. doi: 10.1109/TVCG.2013.124

[5] C. Chen, Y. Liu, S. Kreiss, and A. Alahi. Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning. In *Proc. of ICRA*, pp. 6015–6022. IEEE, New Yrok, 2019. doi: 10.1109/ICRA.2019.8794134

[6] Y. F. Chen, M. Liu, M. Everett, and J. P. How. Decentralized Non-communicating Multiagent Collision Avoidance with Deep Reinforcement Learning. In *Proc. of ICRA*, pp. 285–292. IEEE, New Yrok, 2017. doi: 10.1109/ICRA.2017.7989037

[7] Y. Cheng, Y. Yan, X. Yi, Y. Shi, and D. Lindlbauer. SemanticAdapt: Optimization-based Adaptation of Mixed Reality Layouts Leveraging Virtual-Physical Semantic Connections. In *Proc. of UIST*, pp. 282–297. ACM, New Yrok, 2021. doi: 10.1145/3472749.3474750

[8] X. Chu, X. Xie, S. Ye, H. Lu, H. Xiao, Z. Yuan, C. Zhu-Tian, H. Zhang, and Y. Wu. TIVEE: Visual Exploration and Explanation of Badminton Tactics in Immersive Visualizations. *IEEE Trans. Vis. Comput. Graph.*, 28(1):118–128, 2022. doi: 10.1109/TVCG.2021.3114861

[9] D. Dewey. Reinforcement Learning and the Reward Engineering Principle. In *AAAI Spring Symposia*. AAAI Press, Palo Alto, 2014.

[10] M. S. Esmaeeli and H. Malek. Evolutionary Deep Reinforcement Learning Using Elite Buffer: A Novel Approach Towards DRL Combined with EA in Continuous Control Tasks. *CoRR*, abs/2209.08480, 2022. doi: 10.48550/arXiv.2209.08480

[11] A. Fender, P. Herholz, M. Alexa, and J. Müller. OptiSpace: Automated Placement of Interactive 3D Projection Mapping Content. In *Proc. of CHI*, p. 269. ACM, New York, 2018. doi: 10.1145/3173574.3173843

[12] A. Fender, D. Lindlbauer, P. Herholz, M. Alexa, and J. Müller. HeatSpace: Automatic Placement of Displays by Empirical Analysis of User Behavior. In *Proc. of UIST*, pp. 611–621. ACM, New York, 2017. doi: 10.1145/3126594.3126621

[13] C. Gebhardt, B. Hecox, B. van Opheusden, D. Wigdor, J. Hillis, O. Hilliges, and H. Benko. Learning Cooperative Personalized Policies from Gaze Data. In *Proc. of UIST*, pp. 197–208. ACM, New Yrok, 2019. doi: 10.1145/3332165.3347933

[14] R. Grasset, T. Langlotz, D. Kalkofen, M. Tatzgern, and D. Schmalstieg. Image-driven View Management for Augmented Reality Browsers. In *Proc. of ISMAR*, pp. 177–186. IEEE Computer Society, Los Alamitos, 2012. doi: 10.1109/ISMAR.2012.6402555

[15] J. Grubert, T. Langlotz, S. Zollmann, and H. Regenbrecht. Towards Pervasive Augmented Reality: Context-Awareness in Augmented Reality. *IEEE Trans. Vis. Comput. Graph.*, 23(6):1706–1724, 2017. doi: 10.1109/TVCG.2016.2543720

[16] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proc. of ICML*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, New York, 10–15 Jul 2018. doi: 10.48550/arXiv.1801.01290

[17] S. Hegde, J. Maurya, R. Hebbalaguppe, and A. Kalkar. SmartOverlays: A Visual Saliency Driven Label Placement for Intelligent Human-Computer Interfaces. In *Winter Conference on Applications of Computer Vision*, pp. 1110–1119. IEEE, New York, 2020. doi: 10.1109/WACV45572.2020.9093587

[18] G. Huang, X. Qian, T. Wang, F. Patel, M. Sreeram, Y. Cao, K. Ramani, and A. J. Quinn. AdapTutAR: An Adaptive Tutoring System for Machine Tasks in Augmented Reality. In *Proc. of CHI*, pp. 417:1–417:15. ACM, New York, 2021. doi: 10.1145/3411764.3445283

[19] T. S. Jaakkola, S. Singh, and M. I. Jordan. Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. In G. Tesauro, D. S. Touretzky, and T. K. Leen, eds., *Proc. of NIPS*, pp. 345–352. MIT Press, Cambridge, 1994. doi: 10.5555/2998687.2998730

[20] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, et al. Unity: A General Platform for Intelligent Agents. *CoRR*, abs/1809.02627, 2018. doi: 10.48550/arXiv.1809.02627

[21] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. K. Yogamani, and P. Pérez. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Trans. Intell. Transp. Syst.*, 23(6):4909–4926, 2022. doi: 10.1109/TITS.2021.3054625

[22] T. Köppel, M. E. Gröller, and H. Wu. Context-Responsive Labeling in Augmented Reality. In *Proc. of PacificVis*, pp. 91–100. IEEE, New York, 2021. doi: 10.1109/PacificVis52677.2021.00020

[23] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by Example. *Comput. Graph. Forum*, 26(3):655–664, 2007. doi: 10.1111/j.1467-8659.2007.01089.x

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous Control with Deep Reinforcement Learning. *CoRR*, abs/1509.02971, 2015. doi: 10.48550/arXiv.1509.02971

[25] T. Lin, R. Singh, Y. Yang, C. Nobre, J. Beyer, M. A. Smith, and H. Pfister. Towards an Understanding of Situated AR Visualization for Basketball Free-Throw Training. In *Proc. of CHI*, pp. 461:1–461:13. ACM, New York, 2021. doi: 10.1145/3411764.3445649

[26] T. Lin, Y. Yang, J. Beyer, and H. Pfister. Labeling Out-of-View Objects in Immersive Analytics to Support Situated Visual Searching. *IEEE Trans. Vis. Comput. Graph.*, 29(3):1831–1844, 2023. doi: 10.1109/TVCG.2021.3133511

[27] T. Lin, C. Zhu-Tian, Y. Yang, D. Chiappalupi, J. Beyer, and H. Pfister. The Quest for Omnioculars: Embedded Visualization for Augmenting Basketball Game Viewing Experiences. *CoRR*, abs/2209.00202, 2022. doi: 10.48550/arXiv.2209.00202

[28] D. Lindlbauer, A. M. Feit, and O. Hilliges. Context-Aware Online Adaptation of Mixed Reality Interfaces. In F. Guimbretière, M. S. Bernstein, and K. Reinecke, eds., *Proc. of UIST*, pp. 147–160. ACM, New Yrok, 2019. doi: 10.1145/3332165.3347945

[29] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley. Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review. *Robotics*, 10(1):22, 2021. doi: 10.3390/robotics10010022

[30] J. B. Madsen, M. Tatzgern, C. B. Madsen, D. Schmalstieg, and D. Kalkofen. Temporal Coherence Strategies for Augmented Reality Labeling. *IEEE Trans. Vis. Comput. Graph.*, 22(4):1415–1423, 2016. doi: 10.1109/TVCG.2016.2518318

[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proc. of ICML*, vol. 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937. PMLR, New York, 20–22 Jun 2016. doi: 10.5555/3045390.3045594

[32] P. Mohr, S. Mori, T. Langlotz, B. H. Thomas, D. Schmalstieg, and D. Kalkofen. Mixed Reality Light Fields for Interactive Remote Assistance. In R. Bernhaupt, F. F. Mueller, D. Verweij, J. Andres, J. McGrenere, A. Cockburn, I. Avellino, A. Goguey, P. Bjøn, S. Zhao, B. P. Samson, and R. Kocielnik, eds., *Proc. of CHI*, pp. 1–12. ACM, New York, 2020. doi: 10.1145/3313831.3376289

[33] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *CoRR*, abs/2003.04960, 2020. doi: 10.48550/arXiv.2003.04960

[34] B. Nuernberger, E. Ofek, H. Benko, and A. D. Wilson. SnapToReality: Aligning Augmented Reality to the Real World. In *Proc. of CHI*, pp. 1233–1244. ACM, New York, 2016. doi: 10.1145/2858036.2858250

[35] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training Language Models to Follow Instructions with Human Feedback. In *Proc. of NeurIPS*, vol. 35, pp. 27730–27744. Curran Associates, Inc., Red Hook, 2022. doi: 10.48550/arXiv.2203.02155

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang,

Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. of NeurIPS*, vol. 32, pp. 8024–8035. Curran Associates, Inc., Red Hook, 2019. doi: 10.5555/3454287.3455008

[37] X. Qian, F. He, X. Hu, T. Wang, A. Ipsita, and K. Ramani. ScalAR: Authoring Semantically Adaptive Augmented Reality Experiences in Virtual Reality. In *Proc. of CHI*, pp. 65:1–65:18. ACM, New York, 2022. doi: 10.1145/3491102.3517665

[38] N. Rakholia, S. Hegde, and R. Hebbalaguppe. Where to Place: A Real-Time Visual Saliency Based Label Placement for Augmented Reality Applications. In *Proc. of ICIP*, pp. 604–608. IEEE, New York, 2018. doi: 10.1109/ICIP.2018.8451052

[39] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *CoRR*, abs/1506.02438, 2016. doi: 10.48550/arXiv.1506.02438

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. doi: 10.48550/arXiv.1707.06347

[41] N. O. Service. What is lidar? `https://oceanservice.noaa.gov/facts/lidar.html`.

[42] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao. A Survey of Deep Reinforcement Learning in Video Games. *CoRR*, abs/1912.10944, 2019. doi: 10.48550/arXiv.1912.10944

[43] SportVU. Papers with code - nba sportvu dataset. `https://paperswithcode.com/dataset/nba-sportvu`.

[44] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, Camdbridge, 2018. doi: 10.5555/3312046

[45] T. Tahara, T. Seno, G. Narita, and T. Ishikawa. Retargetable AR: Context-aware Augmented Reality in Indoor Scenes based on 3D Scene Graph. In *Proc. of ISMAR*, pp. 249–255. IEEE Computer Society, Los Alamitos, 2020. doi: 10.1109/ISMAR-Adjunct51615.2020.00072

[46] T. Tang, R. Li, X. Wu, S. Liu, J. Knittel, S. Koch, L. Yu, P. Ren, T. Ertl, and Y. Wu. PlotThread: Creating Expressive Storyline Visualizations using Reinforcement Learning. *IEEE Trans. Vis. Comput. Graph.*, 27(2):294–303, 2021. doi: 10.1109/TVCG.2020.3030467

[47] M. Tatzgern, D. Kalkofen, R. Grasset, and D. Schmalstieg. Hedgehog labeling: View management techniques for external labels in 3d space. In *Proc. of VR*, pp. 27–32. IEEE Computer Society, Los Alamitos, 2014. doi: 10.1109/VR.2014.6802046

[48] M. Tatzgern, V. Orso, D. Kalkofen, G. Jacucci, L. Gamberini, and D. Schmalstieg. Adaptive information density for augmented reality displays. In *Proc. of VR*, pp. 83–92. IEEE Computer Society, Los Alamitos, 2016. doi: 10.1109/VR.2016.7504691

[49] W. Tong, C. Zhu-Tian, M. Xia, L. Y. Lo, L. Yuan, B. Bach, and H. Qu. Exploring Interactions with Printed Data Visualizations in Augmented Reality. *IEEE Trans. Vis. Comput. Graph.*, 29(1):418–428, 2023. doi: 10.1109/TVCG.2022.3209386

[50] A. Vemula, K. Muelling, and J. Oh. Social Attention: Modeling Attention in Human Crowds. In *Proc. of ICRA*, pp. 1–7. IEEE, New Yrok, 2018. doi: 10.1109/ICRA.2018.8460504

[51] A. Wu, W. Tong, T. Dwyer, B. Lee, P. Isenberg, and H. Qu. MobileVisFixer: Tailoring Web Visualizations for Mobile Phones Leveraging an Explainable Reinforcement Learning Framework. *IEEE Trans. Vis. Comput. Graph.*, 27(2):464–474, 2021. doi: 10.1109/TVCG.2020.3030423

[52] L. Yao, A. Bezerianos, R. Vuillemot, and P. Isenberg. Visualization in Motion: A Research Agenda and Two Evaluations. *IEEE Trans. Vis. Comput. Graph.*, 28(10):3546–3562, 2022. doi: 10.1109/TVCG.2022.3184993

[53] S. Ye, C. Zhu-Tian, X. Chu, Y. Wang, S. Fu, L. Shen, K. Zhou, and Y. Wu. ShuttleSpace: Exploring and Analyzing Movement Trajectory in Immersive Visualization. *IEEE Trans. Vis. Comput. Graph.*, 27(2):860–869, 2021. doi: 10.1109/TVCG.2020.3030392

[54] M. Zhou, Q. Li, X. He, Y. Li, Y. Liu, W. Ji, S. Han, Y. Chen, D. Jiang, and D. Zhang. Table2Charts: Recommending Charts by Learning Shared Table Representations. In *Proc. of KDD*, pp. 2389–2399. ACM, New Yrok, 2021. doi: 10.1145/3447548.3467279

[55] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven Visual Navigation in Indoor Scenes Using Deep Reinforcement Learning. In *Proc. of ICRA*, pp. 3357–3364. IEEE, New Yrok, 2017. doi: 10.1109/ICRA.2017.7989381

[56] C. Zhu-Tian, Y. Su, Y. Wang, Q. Wang, H. Qu, and Y. Wu. MARVisT: Authoring Glyph-Based Visualization in Mobile Augmented Reality. *IEEE Trans. Vis. Comput. Graph.*, 26(8):2645–2658, 2020. doi: 10.1109/TVCG.2019.2892415

[57] C. Zhu-Tian, W. Tong, Q. Wang, B. Bach, and H. Qu. Augmenting Static Visualizations with PapARVis Designer. In *Proc. of CHI*, pp. 1–12. ACM, New Yrok, 2020. doi: 10.1145/3313831.3376436

[58] C. Zhu-Tian, Q. Yang, J. Shan, T. Lin, J. Beyer, H. Xia, and H. Pfister. iBall: Augmenting Basketball Videos with Gaze-moderated Embedded Visualizations. In *Proc. of CHI*, pp. 841:1–841:18. ACM, New York, 2023. doi: 10.1145/3544548.3581266