This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729

Reinforced Labels: Multi-Agent Deep Reinforcement Learning for Point-Feature Label Placement

Petr Bobák, Ladislav Čmolík, Martin Čadík

Abstract—Over the recent years, Reinforcement Learning combined with Deep Learning techniques has successfully proven to solve complex problems in various domains, including robotics, self-driving cars, and finance. In this paper, we are introducing Reinforcement Learning (RL) to *label placement*, a complex task in data visualization that seeks optimal positioning for labels to avoid overlap and ensure legibility. Our novel point-feature label placement method utilizes Multi-Agent Deep Reinforcement Learning to *learn* the label placement strategy, the first machine-learning-driven labeling method, in contrast to the existing hand-crafted algorithms designed by human experts. To facilitate RL learning, we developed an environment where an agent acts as a proxy for a label, a short textual annotation that augments visualization. Our results show that the strategy trained by our method significantly outperforms the random strategy of an untrained agent and the compared methods designed by human experts in terms of *completeness* (*i.e.*, the number of placed labels). The trade-off is increased computation time, making the proposed method slower than the compared methods. Nevertheless, our method is ideal for scenarios where the labeling can be computed in advance, and completeness is essential, such as cartographic maps, technical drawings, and medical atlases. Additionally, we conducted a user study to assess the perceived performance. The outcomes revealed that the participants considered the proposed method to be significantly better than the other examined methods. This indicates that the improved completeness is not just reflected in the quantitative metrics but also in the subjective evaluation by the participants.

Index Terms—Point-Feature Label Placement, Machine Learning, Multi-Agent Reinforcement Learning

1 Introduction

Textual *labels* on top of a visualization serve as an additional informational layer (known as *labeling*) to features of interest. An example of such a visualization could be a geographical map where cities (*point features*), highways (*line features*), and bodies of water (*area features*) are the features of interest. However, according to Yoeli [1], a founder of automated label placement, manual labeling of geographical maps takes up to 50% of the overall production time. On the other hand, even simplified versions of the label placement problem are NP-hard [2]. Therefore, it is often necessary to use heuristics in real-world scenarios to find feasible positions for a large number of labels.

The automatic label placement was acknowledged by ACM Computational Geometry Impact Task Force [3] as an important area of research more than two decades ago. In this work, we focus on automatic point-feature label placement (PFLP) or, concisely and interchangeably, point-feature labeling (PFL). Over the past decades, many distinct techniques were published: techniques based on mathematical programming [4], simulated annealing [5], tabu search [6], [7], evolution and genetic algorithms [8], force-based algorithms [9], and rule-based algorithms [10], [11], [12], [13]. These techniques share one commonality – all of them are hand-crafted algorithms designed by human experts. Mathematical programming-based approaches often suffer from extensive computation time, while objectives are challenging to optimize due to solver limitations (*e.g.*, linearity, integral or quadratic constraints), and optimality is only guaranteed in limited cases. As a result, hand-engineered objectives may not describe the actual objective function of PFL in the sense of mathematical formulation. Rule-based techniques are greedy algorithms, that once the solution for a label is found, its position cannot be rearranged later. Therefore, they do not guarantee optimality and are less flexible by design since objectives have to be hardcoded in the algorithm. Evolution, genetic, and force-based approaches often get stuck in local optima.

Just as deep learning vastly mitigated the need for feature engineering and produced remarkable performance improvements in computer vision and natural language processing, applying learning techniques to visualization could lead to similar advancements.

In this work, we present point-feature label placement as a *reinforcement learning* problem. Reinforcement Learning (RL) is an area of machine learning (alongside supervised and unsupervised learning) concerned with decision-making, driven by experience, to maximize the numerical reward signal. Over the discrete-time steps, an agent (*i.e.*, decision-maker) senses the *state* of the *environment*, interacts with the environment by taking *actions* that affect the state, and receives a *reward*. Learning from experience overcomes the lack of label placement datasets that would be needed for supervised learning. Furthermore, the advancement in Deep Learning and its combination with RL emerged in Deep Reinforcement Learning (DRL). DRL has enabled scaling the RL to previously intractable problems (including labeling) due to the curse of dimensionality. Recent research has successfully proven that DRL can solve complex problems within several domains even with enormous state spaces, *e.g.*, robotics [14], [15], self-driving cars [16], industrial design [17], and finance [18]. Moreover, DRL can be trained for objectives that are difficult to optimize directly, as DRL is agnostic to the precise model of the environment as long as the reward signal correlates with the objective. Li and Malik [19]

- *P. Bobák and M. Čadík are with the Faculty of Information Technology, Brno University of Technology. E-mail: {ibobak | cadik}@fit.vutbr.cz.*
- *L. Čmolík is with the Faculty of Electrical Engineering, Czech Technical University in Prague. E-mail: cmolikl@fel.cvut.cz.*

demonstrated that an RL-based autonomous optimization algorithm converges faster and/or finds better optima than the existing hand-engineered optimization algorithms (*e.g.*, gradient descent, momentum, conjugate gradient).

Given the mentioned characteristics, we believe the DRL approach is well suited for the point-feature label placement problem. However, employing RL to label placement poses unique challenges, such as complex deep RL training, handling variable number of anchors and labels, and vast continuous state and action spaces. Moreover, it demands the meticulous design of neural network architectures, formulation of state space, and careful definition of action space. Despite these challenges, the potential of RL in this domain is immense. Our research confronts these issues, devising strategies highlighting our novel approach to the problem.

To the best of our knowledge, no work has been published on label placement with the ability to learn and generalize from experience and generate new labeling for unseen sets of features of interest. Our main contributions are summarized as follows:

(1) We introduce a Multi-Agent Deep Reinforcement Learning formulation to the point-feature labeling problem, which we believe to be the first machine-learning-driven labeling method contrary to the existing hand-crafted algorithms designed by human experts. Our efficient feedforward neural network architecture, with less than half a million parameters, serves as both a policy and value function approximation.

(2) We provide comprehensive ablation studies on the observed modalities and neural network architecture that justify our design choices, highlight the essential aspects of our work, and provide a guideline for future work. As a part, we introduce a novel *completeness metric* that measures the performance of labeling methods based on the number of completely labeled point features and *benchmark dataset* for evaluation.

(3) We compare the performance of the proposed method with two existing methods, Particle-Based Labeling [11] and Rapid Labels [13], using quantitative assessments. Additionally, we conducted a user study to evaluate the proposed method qualitatively. Based on these evaluations, we show that our method outperforms all the other examined methods.

2 Related Work

In our review of related work, we adopt the classification structure presented by Luboschik *et al*. [11]. We first discuss point-feature labeling approaches that utilize the prevalent *fixed-position* model. These techniques make use of a few predetermined candidate locations in close proximity to each point feature and select one of these as the label position. Subsequently, we delve into methods that employ the *slider* model, allowing for more flexible label placement by permitting the label to slide around its associated point feature. Illustrations of the fixed-position and slider models can be found in Figure 1. The labels in fixed-position and slider models are denoted by Luboschik *et al*. [11] as *adjacent* labels as they are positioned adjacent to the point feature. Only a few approaches [11], [20] have gone a step further to enhance the flexibility of the fixed-position or slider model by introducing *distant labels*. These labels are permitted to be placed further away from the associated point features while the association is maintained via lines or curves. In the following text, we explicitly mention which bodies of work allow distant labels. As our work harnesses reinforcement learning to address the point feature labeling problem, we also explore visualization approaches that leverage machine learning.

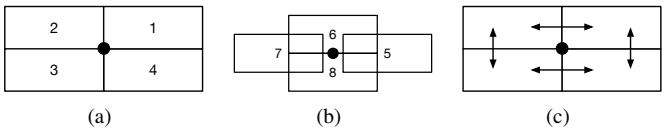


Fig. 1. Illustration of label placement models according to Luboschik *et al.* [11]. Image (a) depicts the 4-position model, (b) shows the extended positions of the 8-position model, and (c) displays the slider model. The number of each label position indicates the rank where one is the most preferred position and eight is the least preferred one.

2.1 Fixed-position Models

The methods based on fixed-position models are restricted in terms of possible label positions, and thus the search space is considerably narrowed. The goal is to place labels for as many point features as possible by choosing from the pre-determined candidates. Often, rules are applied to simplify the resolution of conflicts between the candidates [21]. The optimal label layout (*i.e.*, the label layout with the maximum number of placed labels) is often not achieved because the optimal position of labels is not present in the discrete space of the fixed-position model. Nevertheless, the fixed-position models have been studied more extensively than the slider models.

Various optimization strategies have been employed to find the label placement. Zoraster [4] proposed an integer linear programming formulation of label placement in maps specific to the oil industry. Christensen and Marks [5] and Zoraster [20] proposed methods based on simulated annealing. The approach of Zoraster expands the fixed-position model to incorporate distant labels. Each distant label is chosen from candidates that are located in the same directions from the point feature as in the fixed-position model but at two additional distance levels. Yamamoto *et al*. [6] introduced the combinatorial optimization method based on tabu search. Alvim *et al*. [7] proposed a point-feature labeling method based on the application of the POPMUSIC (Partial Optimization Metaheuristic under Special Intensification Conditions). The POPMUSIC divides the PFL into subproblems that are optimized with tabu search. Rabello *et al*. [22] presented a clustering search metaheuristic, which identifies promising search areas while avoiding over-exploring poor or well-explored regions. Lu *et al*. [8] proposed a hybrid approach combining discrete differential evolution and genetic labeling algorithm for all types of features (*i.e.*, point, line, area). While all the above-described methods provide high-quality label layouts, they share a common problem. They require a significant computational effort, which results in high runtimes.

Unlike optimization methods, greedy methods reduce the runtime by sequentially labeling the point features without the ability to recover from bad solutions or with a limited recovery step at the end [23]. Typically, greedy methods will find a worse label layout but in a significantly shorter time (milliseconds) which makes them usable in time-critical applications. Mote [10] introduced a greedy method that divides the screen into a 2-dimensional grid and determines a simplified version of a conflict graph of the point features. Only point features in neighboring cells need to be checked for conflict with a given point feature. Luboschik *et al*. [11] proposed a greedy particle-based labeling (PBL) approach that can also respect other visual elements and the visual extent of labeled features using conflict particles representing the occupied space. Each placed label is approximated by a set of newly generated particles. The performance of PBL depends heavily on the number of used particles. The method is significantly

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729

slower for cases with complex visual features represented by many particles. Kittivorawong *et al*. [12] addressed the performance problem of particle-based labeling with an occupancy bitmask that allows faster evaluation of label overlaps with complex visual features that cannot be occluded. Nevertheless, the computation time required to determine overlap for a label still depends on the size of the label and the resolution of the screen. Recently, Pavlovec and Čmolík [13] introduced Rapid labels, an approach leveraging the power of GPU. They are allowing to label several point features in each iteration using a 2-dimensional grid. Further, they evaluate the overlaps of labels with important visual elements, the conflicts between the labels, and the ambiguity of the labels to position the labels in suitable order. They utilize Summed Area Table to evaluate the overlaps, conflicts, and ambiguity independently of label size and screen resolution.

2.2 Slider Models

Compared to fixed-position models, slider models provide greater label placement flexibility by allowing continuous label movement around their corresponding features. However, slider models have not received much attention from the scientific community, and only a few approaches have been proposed. For instance, Hirsh [9] proposed a forced-based algorithm, where the translation vectors are repeatedly computed based on repelling forces for overlapping labels. Doddi *et al*. [24] introduced a slider model where the goal is to maximize the size of square or circle-shaped labels. The presented method allows rotating each label around the corresponding feature. Kreveld *et al*. [25] studied several sliding models (top-, two- and four-slider models) and proposed a combinatorial polynomial-time greedy approximation approach for all mentioned slider models. Li *et al*. [26] proposed a method rooted in the plane collision detection theory, using the region of movability to define a conflict-free search space for label placement and determine the best label positions through heuristic search. The mentioned approach of Luboschik *et al*. [11] also utilizes the slider model, sequentially positioning labels using fixed-position, slider, and distant label models with straight-line leaders. For distant labels, they sample candidate positions from a spiral function. Their approach supports interactive labeling for up to 1000 features.

2.3 Machine Learning & Visualization

There has been a recent trend of incorporating machine learning techniques into visualization (ML4VIS) to enhance the efficiency of visualization creation and suggest appropriate visual representations. Machine learning can be broadly categorized into several types, including supervised learning and reinforcement learning, each with its unique characteristics and suitable application areas. Supervised learning is the most common type, where an algorithm learns a model from labeled training data and then uses this model to make predictions for unseen data. In the context of visualization, supervised learning has been utilized in several impactful ways. For instance, Luo *et al*. [8] introduced DeepEye, a system that leverages supervised learning to recommend suitable visualizations based on the provided data. Likewise, Bylinskii *et al*. [27] formulated a method for predicting users' visual attention distribution on infographics. Similarly, Chen *et al*. [28] employed a series of supervised-learning techniques for the automatic segmentation of graphical elements from timeline infographics.

Reinforcement learning represents another significant category of machine learning, where an agent learns to make decisions by taking actions in an environment to maximize some notion of cumulative reward. Despite the success of reinforcement learning across various domains, its application in visualization (RL4VIS) remains relatively unexplored, with only a handful of works adopting this technique. For instance, Tang *et al*. [29] proposed PlotThread, which facilitates collaboration between an RL agent and a human designer to modify the storyline's layout. The trained agent assists designers and refines their interactions on a shared canvas. Zhou *et al*. [30] proposed Table2Chart leveraging deep Q-learning and heuristic search to generate chart sequences from table data. Hu *et al*. [31] proposed a method for optimizing the coordinate ordering of sets of star glyphs related to multiple class labels to maximize perceptual class separation. In addition, Wu *et al*. [32] developed MobileVisFixer, which automates a mobile-friendly visualization re-design process, and Deng *et al*. [33] introduced DashBot, a Deep RL-based tool for generating analytical dashboards. For a more comprehensive review of the use of machine learning in visualization, please refer to the survey by Wang *et al*. [34]. While RL provides innovative solutions for visualization problems, visual analytics also proves instrumental in understanding and interpreting RL models (VIS4RL). For instance, DQNViz [35] proposed by Wang *et al*. and DRLViz [36] introduced by Jaunet *et al*. offer visual analytics approaches to understand deep Q-networks and deep reinforcement learning, respectively. Mishra *et al*. [37] visualized explanations of agent behavior in reinforcement learning, whereas Wang *et al*. [38] employed visual analytics for RNN-based deep reinforcement learning.

Although the techniques above tackle various tasks, none of them address the specific challenge of label placement. Furthermore, RL-based techniques predominantly rely on a single-agent approach, a strategy that diverges from the multi-agent system proposed in our research. Our work introduces a novel approach wherein multiple agents, acting as proxies for labels, are tasked to learn and interact concurrently within a shared environment. This paradigm inherently involves more complex dynamics, such as agent coordination and management of a non-stationary environment, which arise due to the simultaneous learning processes undertaken by the agents [39], [40]. Therefore, our work represents not only an advancement in the application of reinforcement learning to the intricate problem of label placement but also the exploration of multi-agent systems in the visualization domain.

3 Introduction to Reinforcement Learning

Instead of relying on labeled data common for supervised learning, reinforcement learning (RL) leverages an agent that learns to take actions that maximize a cumulative reward by exploring the environment and observing the consequences of its actions. Reinforcement learning techniques have become an essential tool for solving various complex problems, particularly when dealing with decision-making. However, the curse of dimensionality in state space makes it challenging to apply RL effectively. To overcome this, RL is usually extended to Deep Reinforcement Learning (DRL), which employs a neural network as a function approximator [41]. Based on the number of agents, RL can be further classified into a Single-Agent Reinforcement Learning (SARL) variant, where only one agent interacts with the environment. The second variant is Multi-Agent Reinforcement Learning (MARL), where multiple agents interact with a shared environment and work collaboratively to attain a shared goal.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729

The formal basis of reinforcement learning is the theory of Markov Decision Processes (MDP). Over the discrete-time steps $t \in \mathbb{N}$, an agent observes the *state* $s_t \in S$ of the *environment*. Given the state, the agent selects *actions* $a_t \in A$ based on a *policy* π that maps the state to a probability distribution over actions. At the next time step $t+1$ as a consequence of the selected action, the agent receives feedback in a form of numerical *reward* r_{t+1} and transitions to new state s_{t+1}.[1] The sequence of states, actions, and rewards is called *trajectory* or *rollout*. The agent's goal is to find an *optimal policy* π_* that maximizes the *discounted return* $G_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$, where $\gamma \in [0,1]$ is a *discount factor*. As $\gamma \to 0$, the agent becomes more shortsighted and maximizes the immediate reward. On the other hand, the agent takes the future rewards into account more seriously when $\gamma \to 1$. The agent's task is called *episodic* if $T < \infty$, and such a trajectory is called an *episode*. The *value function* $v_\pi(s)$, also called the state-value function, is the expected return beginning in state s and following policy π afterward. Formally, the value function is defined as $v_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$. Likewise, the *q-value function* $q_\pi(s,a)$, also called action-value function, is the expected return beginning in state s, taking action a, and following policy π afterward. Formally, the value function is defined as $q_\pi(s,a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$. In the DRL, the Markov property is often relaxed so that the agent does not have to be fully aware of the environmental state. Instead, the agent observes only a partial state known as *observation* o_t (retains far less information compared to the entire state of the environment), forming the Partially Observable MDP (POMDP). One step further is the cooperative Multi-Agent setting, where the agents act in parallel, extending the POMDP to the Decentralized POMDP (dec-POMDP). We refer to Sutton and Barto [41] for further details on the formal background.

4 Learning Point-feature Label Placement

Traditionally, supervised learning methods require a substantial amount of labeled data samples to train the model. However, this requirement poses a significant challenge, as no such labeled dataset currently exists for the point-feature label problem. Creating a dataset for this purpose would require access to an immense number of high-quality drawings with labels and time-consuming annotations. The novel approach presented in this work tackles this problem by posing the point-feature label placement as a reinforcement learning problem, which circumvents the challenge posed by the absence of ground-truth label placement datasets, a crucial requirement for traditional supervised learning methods.

Designing the labeling problem as MARL has several significant benefits over SARL. First, when posing the label placement as SARL, the agent acts as a supervisor managing all labels at once. However, in that case, the observation and action space size changes with a varying number of labels, which contradicts the RL's prerequisite of fixed-sized observation and action spaces. On the contrary, with the abstraction of an agent for each label in MARL, an agent's individual observation and action space can be implicitly designed as fixed-sized. Therefore, the number of agents (*i.e.*, copies of trained strategy) varies in MARL compared to the observation and action space size in SARL. Second, even if one would overcome the variability, the SARL observation space would still be several times larger than in MARL, and the neural network architecture would be more complex and have significantly more parameters, resulting in more challenging training. Third, one would have to collect many more trajectories to train the super agent, as the trajectory in SARL is a collection of the observations, actions, and rewards of all labels. On the contrary, the trajectory of each individual agent can be used to improve the shared strategy in MARL. Therefore, due to the mentioned properties, we decided to represent each label by an agent, which finally unfolds into a Multi-Agent Deep Reinforcement Learning (MADRL) problem. From now on, we also refer to an agent as a label or a label agent interchangeably. Similarly, we refer to a point feature as an *anchor*.

The proposed method is designed explicitly for *adjacent* PFL, meaning a label can be placed only around its anchor. Our intention is to find a conflict-free label position for each anchor (denoted as *complete* labeling), and if such a position does not exist for all or cannot be found by the method, we call the labeling *incomplete*. The emergence of label placement strategy and rules, in RL referred to as a policy, of our method is driven entirely by the learning process contrary to the existing hand-crafted methods.

4.1 Environment

In the following text, we follow the terminology and definitions defined by Bekos *et al*. [42], later extended by Bobák *et al*. [43].

Inspired by the interface of OpenAI Gym [44], we transformed the adjacent PFL problem into a custom-developed environment, referred to as `AdjacentPFLEnv`, to facilitate the reinforcement learning paradigm. The proposed environment consists of a set of anchors $\mathscr{A}$, each defined by its coordinates $(a_\mathrm{x}, a_\mathrm{y})$ enclosed within rectangular *drawing region* D of dimensions $(D_\mathrm{w}, D_\mathrm{h})$. Each anchor is paired with an axis-aligned box denoted as *label agent* ℓ defined by its *origin* coordinates $(\ell_\mathrm{x}, \ell_\mathrm{y})$ and dimensions $(\ell_\mathrm{w}, \ell_\mathrm{h})$. A set of all label agents within the environment is denoted as $\mathscr{L}$. The label agent's origin coordinates lie on the circumference of the *slider rectangle* σ, whose origin is defined as $(\sigma_x, \sigma_y) = (a_x - \ell_w, a_y - \ell_h)$ and dimensions as $(\sigma_w, \sigma_h) = (\ell_\mathrm{w}, \ell_\mathrm{h})$. Finally, each label agent ℓ remains tethered to its respective anchor via an attachment point denoted as *port* Π, see Figure 2(a). We derive the initial origin of an associated label agent as

$$\ell_x = \mathrm{clip}(\mathrm{a_x}, 0, \mathrm{D_w} - \ell_\mathrm{w}), \tag{1a}$$

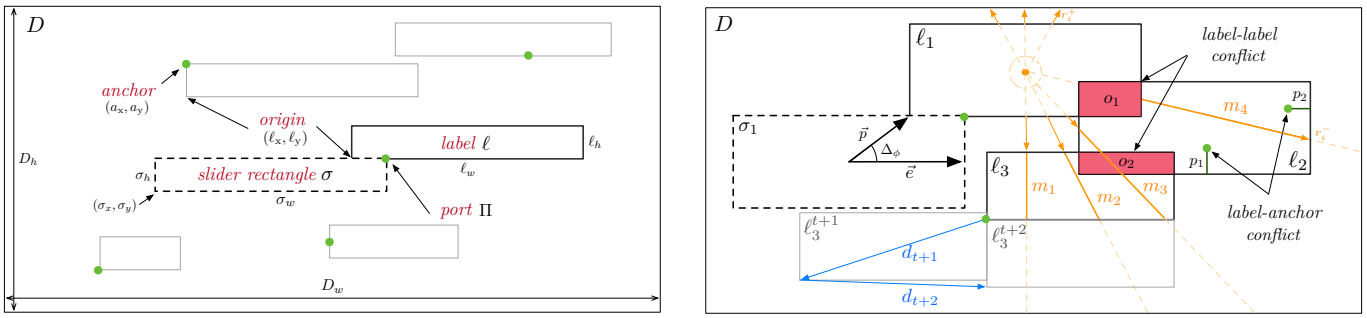
$$\ell_y = \mathrm{clip}(\mathrm{a_y}, 0, \mathrm{D_h} - \ell_\mathrm{h}), \tag{1b}$$

where $\mathrm{clip}(x, b_l, b_u)$ is a piecewise function that clips the value x between lower b_l and upper b_u bound. Therefore, a label agent is placed at the initial state s_0 primarily at the most preferred position in the upper right quadrant of the 4-position model, despite the fact that agents can be in *conflict* (*i.e.*, agent-agent overlap, agent-anchor penetration), see Figure 2(b).

In the training phase, the anchor coordinates and dimensions of associated label agents are randomized at the initial state s_0. Coordinates of an anchor are drawn from the uniform distribution such as $a_x \sim U(0, D_w)$ and $a_y \sim U(0, D_h)$. Dimensions of label agents are determined in the similar fashion $\ell_w \sim U(0.1D_w, 0.15D_w)$, and we fixed $\ell_h = 0.05D_h$. The environment is populated only by one to two label agents, none of which or both of which overlap with the other agent, and we chose $D_w = 600$, $D_h = 400$ pixels. The environment terminates at the fixed horizon of $T = 100$ steps. After termination, a new randomized configuration of label agents is populated in the environment and presented to the training algorithm. We apply a fixed-horizon approach to avoid a non-episodic behavior (*i.e.*, infinite horizon) when conflict-free label placement does not exist. Furthermore, a fixed horizon helps

1. We follow the notation of Sutton and Barto [41] and use the r_{t+1} instead of r_t to denote that reward and next state s_{t+1} are determined jointly.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729



(a) (b)

Fig. 2. Illustration (a) describes the used terminology and definitions. Illustration (b) depicts the observed modalities and defined action Δ_ϕ. The orange dashed lines represent rays providing the mapping vector of modalities. A ray begins at the label's center, but the actual reading of range starts at the label's bound. We collect the distance and type of the nearest intersection (*i.e.*, label, anchor, or bound) for each ray. The distance is positive (denoted by the plus sign subscript) if the ray hits the label from the outside. On the other hand, the distance is negative if the ray hits the label from the inside; see the ray denoted by r_i^-. Furthermore, we compute the mass of bodies of labels (denoted by the $m_{1\ldots4}$) that the ray went through till the bound of the environment. The overlap modality is depicted by the area of o_1 and o_2. The value for ℓ_1 is just the area of o_1, but o_1+o_2 for ℓ_2 as it is in a label-label conflict with ℓ_2 and ℓ_3. Similarly, we demonstrate the penetration distance for ℓ_2 as a sum of p_1 and p_2. Finally, we show the displacement distance d_{t+1} and d_{t+2} time steps $t+1$ and $t+2$, respectively. The total distance traveled is the cumulative sum of d_t, $t \in [0,T]$. The illustration does not show a complete description of the state for the illustration clarity (*e.g.*, only a few rays are visualized, and the labels corresponding to anchors producing the penetration p_1 and p_2 are not shown).

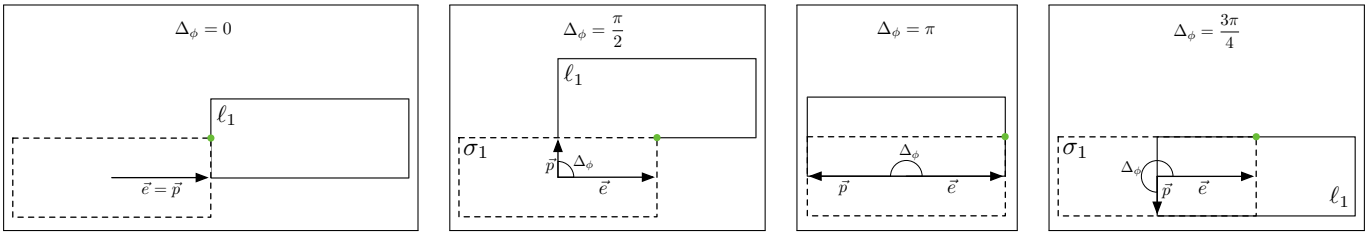


Fig. 3. A detailed illustration of the agent's action Δ_ϕ. For $\Delta_\phi : 0 \to 2\pi$, the origin of the label $[\ell_x,\ \ell_y]$ moves counterclockwise along the circumference of a slider rectangle σ (denoted by a dashed line), whose origin is defined as $(\sigma_x, \sigma_y) = (a_x - \ell_w, a_y - \ell_h)$ and dimensions as $(\sigma_w, \sigma_h) = (\ell_\mathrm{w}, \ell_\mathrm{h})$.

stabilize the label agent's position after finding a conflict-free arrangement by allowing the agent to discover that any additional action can lead to a deterioration of the reward.

In the evaluation phase, we can populate the environment with any number of label agents, even though we trained the policy with only one to two agents. We are leveraging the capability of RL to generalize to *instances* (*i.e.*, specific configurations of an environment) unseen during the training to find the label placement for hundreds of anchors rather than just two. Moreover, the size of the drawing area, label size, and other mentioned parameters can be selected arbitrarily.

4.2 Observation Space

We represent the state of the environment for each label agent ℓ solely by the local same-shaped *observation vector* o_t^ℓ. This approach enables us to leverage all individual trajectories to train a shared policy and facilitate decentralized execution. Furthermore, we solely rely on sensor-based data, raw data acquired directly from the environment, instead of image data. Image data, represented as raster images or pixel matrices, typically demand greater storage and computational resources. By opting for sensor-based data, we effectively reduce the size of the observation vectors. To capture the semantics of the observed modalities, we divided the observation o_t^ℓ into two distinct vectors: the *mapping vector* M and the *self-aware vector* S. The observation vector o_t^ℓ is then obtained by concatenating M and S.

The modalities captured by the mapping vector $M = [\mathtt{d}, \mathtt{t}, \mathtt{c}, \mathtt{m}]$ serve to encode the agent's surroundings through the use of 32 ray sensors, which are uniformly distributed around the label boundaries and function in a similar manner to LiDAR sensors. These sensors measure the distance `d` to the closest intersection point, as well as the type `t` of an object that the ray intersects (*i.e.*, label, anchor, bounds of the environment). Additionally, the mapping vector captures the count `c` and mass `m` of the labels that the ray passed through.

The self-aware vector $S = [\mathtt{O}, \mathtt{D}, \mathtt{Ape}, \mathtt{Apr}, \mathtt{Ad}, \mathtt{T}]$ supplies local modalities that pertain mainly to the agent's conflicts, including overlaps with other agents and penetrations with anchors. We define the overlap modality `O` as a sum of the occluded area between the given agent and the other agents being in conflict. The displacement `D` represents the Euclidean distance of the agent's origins between two consecutive time steps t and $t+1$. Furthermore, we define penetration distance as the Euclidean distance between the penetrated anchor and the nearest point of escape on the circumference of the label. Like the overlap modality, we define the penetration modality `Ape` as a sum of the penetration distances between a given agent and its anchor or the other agents' anchors being in conflict. For both modalities, we also provide the agent with a count of conflicts relative to the total number of label agents. Additionally, the agent observes the Euclidean distance to its anchor `Ad`, the relative position of the anchor to the label agent's port `Apr`, and finally, the information about elapsed time steps `T`.

We normalize all observation modalities. An overview of the observed modalities is presented in Figure 2(b), while a comprehensive description is available in the supplementary material[2].

4.3 Action Space

We designed the action space of agents such that they can change the origin continuously without discretization to the environment's raster, allowing subpixel precision and independence of the drawing area dimensions. Therefore, we define a continuous action Δ_ϕ representing the angle between basis vector $\vec{e} = (1,0)$ and vector $\vec{p} = (p_x, p_y)$, where point $[p_x, p_y] \equiv [\ell_x,\ \ell_y]$ belongs to the circumference of the slider rectangle for a given label ℓ, see Figure 2(b) and Figure 3. We use one-dimensional action space Δ_ϕ rather than an intuitive two-dimensional action space of Δ_x and Δ_y to simplify the learning process. In our experience, multi-dimensional action can introduce unnecessary coordination complexity. Given $\Delta_\phi \in (0, 2\pi)$, we define the origin of a label in quadrant Q1 as

$$[\ell_x,\ \ell_y]_{t+1} = \begin{cases} \frac{\ell_w}{2}[1,\ \tan(\Delta_\phi)] & 0 \leq \Delta_\phi < \arctan\left(\frac{\ell_h}{\ell_w}\right) \\ \frac{\ell_h}{2}[\cot(\Delta_\phi),\ 1] & \arctan\left(\frac{\ell_h}{\ell_w}\right) \leq \Delta_\phi < \frac{\pi}{2}. \end{cases} \tag{2}$$

The derivation for quadrants Q2 to Q4 is trivial – the idea is to transform the Δ_ϕ into the Q1 and then flip the position to the corresponding quadrant. For clarity, we omitted use of $\Delta_\phi \in [-1, 1]$ in our implementation; nevertheless, the previous declaration holds.

To facilitate continuous action space with an infinite number of actions, we define a policy as a parameterized Gaussian distribution. Therefore, instead of learning the probabilities of all possible actions, which is infeasible, we learn statistics of the distribution. Formally, we define policy as

$$\pi_\theta(\Delta_\phi | s_t) = \mathcal{N}\left(\mu_\theta(s_t), \sigma^2_\theta(s_t)\right), \tag{3}$$

where s_t is the current state at time step t, $\mu_\theta(s_t)$ and $\sigma^2_\theta(s_t)$ is the mean and variance of the distribution parametrized by the neural network parameters $\theta \in \mathbb{R}^d$ further described in Subsection 4.5.

4.4 Reward

Our goal in this work involves the emergence of cooperation among agents to find a conflict-free label position for each anchor. Therefore, we combine two types of rewards as suggested by Nguyen *et al*. [45]. The *local* reward r^{local} assigns an agent its feedback solely based on individual efforts, while the *global* reward r^{global} gives an agent its feedback based on the entire state of the environment.

As a local reward, the agent receives step-wise penalization (*i.e*., negative reward) for being in label-label conflict with another agent, meaning both label agents overlap with each other. We compute the *overlap value* $o(\ell)$ in the same way as the overlap modality, as a sum of the occluded area between a given agent and the other agents in conflict.

The global reward is the composition of local rewards among individual agents. Formally, we define the local reward r^{local} and global reward r^{global} as

$$r^{\text{local}}_{t+1}(\ell) = -o(\ell) \qquad r^{\text{global}}_{t+1} = \sum_{\ell \in \mathscr{L}} r^{\text{local}}_{t+1}(\ell), \tag{4}$$

where t is the current time step, $o(\ell)$ is the overlap value. Based on these definitions, we define the total reward for a label agent ℓ as

$$r^{\text{total}}_{t+1}(\ell) = (1-w) \cdot r^{\text{global}}_{t+1} + w \cdot r^{\text{local}}_{t+1}(\ell). \tag{5}$$

By observing the influence of various weight values on the final reward of the trained policy, we recommend using $w = 0.5$.

4.5 Policy & Value Network Architecture

We designed an efficient yet straightforward feedforward network with just less than half of a million (412 thousand) parameters, as depicted in Figure 4. The architecture consists of two input heads – *mapping* and *self-aware*, and two output branches – *value* and *policy*. The ray observations from sensors constituting the mapping head are first passed through a circular 1D-convolution layer proposed by Schubert *et al*. [46] to capture the correlation between individual readings, including the borders of the tensor. The intermediate representation is reshaped to form a 1D tensor. The other observation modalities forming the self-aware head are concatenated and embedded by a dense layer. The mapping and self-aware heads are concatenated and passed by a final shared dense layer. At the end of the architecture, two separate dense layers split the outcome into the value and policy branches.

4.6 Training

We adopt a *parameter sharing* in which each agent utilizes the same policy network with identical parameters. This approach allows us to optimize the parameters of the proposed network using the trajectories of the individual agents. We update the parameters of the policy network in a *policy gradient* fashion – meaning the parameters are updated based on the gradient of an estimate of expected return with respect to the policy parameters. In particular, we utilize Proximal Policy Optimization (PPO), one of the most prominent actor-critic policy gradient methods proposed by Schulman *et al*. [47]. The method is best known for its relative simplicity while preserving the convergence properties of more complex predecessors. We refer to the survey of Arulkumaran *et al*. [48] for further details on the DRL algorithms. The goal is to optimize the parameters θ that maximize the expected discounted return $\mathbb{E}[G_t]$. The surrogate objective of PPO is defined as

$$L(\theta)^{CLIP} = \mathbb{E}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right], \tag{6}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$ is the probability ratio of the new and old policies. Value of $\hat{A}_t$ is the estimator of the *advantage function*, computed by Generic Advantage Estimation introduced by Schulman *et al*. [49], describing whether choosing action a_t in state s_t is better or worse than the average action of policy π. One of the essential concepts of PPO is the clip operation on top of the probability ratio $r_t(\theta)$ that discourages the policy from dramatically changing between training iterations, resulting in convergence issues. Lower values of ε correspond to more consistent policy improvements. On the other hand, higher values yield greater variance and volatility of convergence. The value network objective is formulated as regression via mean square error as

$$L^{VF}(\theta) = \mathbb{E}\left[(v_\pi(s_t|\theta) - v^{\text{target}})^2\right], \tag{7}$$

where $v^{\text{target}} = r_t + \gamma r_{t+1} + \ldots + \gamma^{T-t} v_{\pi_{\theta old}}(s_t)$. Because the proposed neural network architecture shares parameters between policy and value networks, we combine the aforementioned objectives to a composed loss to train both networks simultaneously as $L(\theta) = L(\theta)^{CLIP} - L^{VF}(\theta)$. A detailed list of hyperparameters is available in the supplementary material.

2. Supplementary material related to this project can be accessed via our project page at http://cphoto.fit.vutbr.cz/reinforced-labels.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729

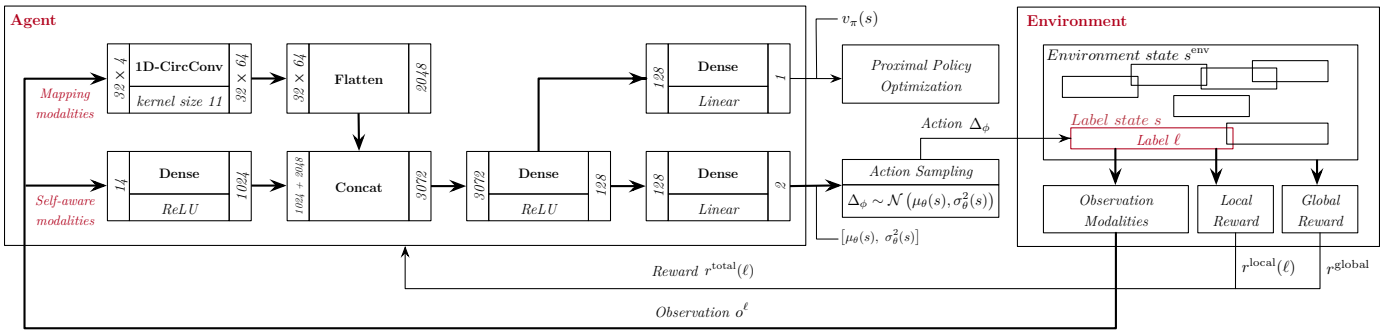


Fig. 4. Overview of Reinforced Labels and its neural network architecture. The proposed `AdjacentPFLEnv` environment describes the adjacent point-feature labeling problem. We are proposing a Multi-Agent approach, where an agent controls the origin of a label. We use parameter sharing, meaning each agent acts according to the same policy network with identical parameters. Therefore, we can use trajectories of individual agents to optimize the parameters of the proposed network. We utilize only local agent observations o^ℓ, enabling parallel execution in the evaluation phase. We provide the agent feedback by local rewards r^{local} and global rewards r^{global} only during the training; the reward is not used in the evaluation phase. The mapping and self-aware modalities are passed through the network, resulting in a state value $v(s)$ and parameters of normal distribution $\mu_\theta(s)$ and $\sigma^2_\theta(s)$. The training algorithm [47] uses the $v(s)$ to optimize the parameters of the network. The distribution parameters are used to sample an action Δ_ϕ. Finally, the agent acts according to the action, which might translate (*i.e.*, the $\Delta_\phi \neq 0$) into the change of the label's origin.

4.7 Inference

Once the policy network is trained, we can use it to evaluate an instance with an arbitrary number of anchors, size of labels, and dimensions of the drawing area, all due to the proposed MADRL design properties and choice of the observed modalities and actions. At the beginning of the evaluation, we initialize the environment according to Equation 1. The evaluation step for an agent ℓ consists of a collection of observation modalities to form vector o^ℓ_t and passing it through the shared policy network, resulting in action Δ_ϕ. This sequence produces a new environment state s_{t+1} and a new observation vector o^ℓ_{t+1}. The process is repeated for all agents until no conflicts are present or the horizon T is hit.

5 Ablation Study

We conducted an ablation study to fine-tune the proposed method and validate our design choices. The goal is to shed light on the impact of each component within our method and to understand how variations in these components can influence overall performance. Additionally, the findings from these ablations open up the potential for future research pathways, creating opportunities for further optimization and refinement of our technique. To measure the performance of our method under different ablations, we introduce a novel metric and benchmark dataset.

5.1 Completeness Metric

Labeling metrics typically measure the quantity of non-conflicting labels. Such metrics, however, are best suited for methods that ensure conflict-free label layouts. Conversely, methods not offering such assurances may calculate performance based on the sum of overlapping regions. Yet, from the perspective of the established label placement rules [1], [50], any overlap, whether slight or significant, deems a layout non-conflict-free. Given this landscape, we observed a gap: while conflict-free methods may omit labels to avoid conflicts, it is uncertain how the overlap regions would appear. Conversely, methods without a conflict-free guarantee do not indicate labels that should be discarded to achieve a conflict-free layout. To close the gap, we introduce the *completeness* metric.

Our work focuses on finding a conflict-free label position for each anchor. If such a position does not exist for all anchors or cannot be found by the method, we call the labeling *incomplete*. On the other hand, we denote the labeling as *complete* if all anchors are annotated without conflict.

We measure the performance of label placement methods by the completeness metric representing the percentage of complete labelings for a given set of instances. For example, let D be a dataset of 10 instances. Let M_1 be PFL method that found eight complete conflict-free layouts without the need of removing any label and two incomplete conflict-free layouts with several removed labels. Let M_2 be a method with the same PFL properties as M_1 that found nine complete conflict-free layouts and one incomplete layout with remaining conflicts (*i.e.*, label-label or label-anchor conflict). The completeness of M_1 is 80%, and the completeness of M_2 is 90%, as the latter method found more complete layouts out of a set of ten given instances. Therefore, method M_2 performs better than method M_1.

5.2 Benchmark Dataset

We have created a benchmark dataset to compare outcomes and performance among the evaluated label placement methods. We split the dataset into two parts – *compact* and *volume* datasets. To endorse the standardized evaluation of labeling methods, we provide the benchmark dataset as supplementary material.

We generated the anchor coordinates using a pseudo-random number generator with uniform distribution common across multiple bodies of previous work [6], [13], [26]. In the compact dataset, we sampled the anchor coordinates from an area of 600×400 and sequentially increased the number of drawn samples by five, ranging from 5 to 50. For the volume dataset, we drew coordinates from an area of 2400×1600 and consecutively raised their count by 50, going from 100 to 600. For each number of anchors, we generated ten instances. Therefore, the entire dataset consists of 41 250 anchors divided into 210 instances. Furthermore, we randomly formed the corresponding labels so that the text consists of three to seven capital letters from the English alphabet.

It is worth noting that the benchmark dataset also contains instances that cannot be solved, as it was created using a pseudo-random number generator, due to the factual inexistence of a conflict-free layout. In such a case, any labeling method cannot produce complete labeling. However, such instances do not influence the comparison of labeling methods on the proposed dataset because these instances always affect all methods the same.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729

TABLE 1
Ablations of neural network architecture evaluated on the compact dataset. We computed the overall completeness metric for the dataset to summarize the performance and simplify the comparison of the ablations examined. Furthermore, we provide the average number of elapsed steps needed to solve instances in the dataset, inference time per step, and the number of parameters to illustrate the complexity of the neural network. The notation is explained in Subsection 5.3

Model	*Completeness* %	*Steps* -	*Inference* ms	*Parameters* millions
RFL_Conv	**97.9**	**30.721**	**3.077**	**0.412**
RFL_ConvLn	96.8	37.333	3.513	0.412
RFL_2Dns	**96.4**	**41.466**	**2.775**	**0.673**
RFL_2DnsLn	94.5	59.353	2.998	0.673
RFL_1Dns	**95.4**	**53.081**	**2.665**	**0.833**
RFL_1DnsLn	94.7	63.307	2.882	0.833
RFL_rllib	90.6	81.905	2.008	0.206

5.3 Architecture Ablations

In order to justify the design choices behind our proposed architecture, we performed several architectural ablations. Each ablation was evaluated on the compact dataset ten times to adequately account for the inherent stochastic properties of Reinforced Labels (RFL). In an effort to summarize the performance and facilitate the comparison of the examined ablations, we report the overall completeness metric as a single average value for the compact dataset. The results of these evaluations can be found in Table 1.

We categorized the ablations into four groups, starting with the proposed architecture and systematically removing/varying its components. Furthermore, we also evaluated the universal baseline architecture of a leading reinforcement learning library RLLib utilized in our development [51]. The first ablation group **RFL_Conv*** contains a custom two-head, two-branch convolution-based architecture as described in Subsection 4.5. The mapping modalities are first concatenated into vector M, embedded by a circular 1D-convolution layer, and optionally passed through a layer normalization. The self-aware modalities are concatenated into vector S, embedded by a dense layer, and optionally passed through a layer normalization denoted by **Ln** suffix. Ultimately, the outcomes are concatenated, and two separate dense layers split the outcome into the value and policy branches. The second ablation group **RFL_2Dns*** retains the two-head, two-branch architecture but excludes the circular 1D-convolution layer. Instead, the mapping modalities are embedded by a dense layer. The third ablation group **RFL_1Dns*** includes custom single-head, two-branch architecture. The observed modalities M and S are merged, embedded by two dense layers, and optionally passed through a layer normalization. At the end of the architecture, two separate dense layers split the outcome into the value and policy branches. The final ablation **RFL_rllib** holds the default baseline architecture of RLLib. The observed modalities M and S are concatenated and separated into the value and policy by two triple-dense-layer branches.

Table 1 shows the results of the architectural ablations. The best-performing model is the RFL_Conv, as described in Subsection 4.5, achieving an overall completeness of 97.9%. We emphasize the importance of the circular 1D-convolution layer as, in addition to classical convolution, the circular one captures the correlation between the borders of the tensor. The second-best score was achieved by RFL_2Dns, without circular convolution but preserving the two-head scheme, achieving 96.4%.

TABLE 2
Ablations of observed modalities evaluated on the compact dataset. We provide overall completeness, the average number of elapsed steps needed to solve the dataset, inference time per step, and observation creation time per step. The notation is explained in Subsection 5.4

Set	Modalities	*Completeness* %	*Steps* -	*Inference* ms	*Observation* ms
1	M008[d cm]S[O ApeAprAdT]	96.8	36.791	2.723	19.870
2	M016[d cm]S[O ApeAprAdT]	96.8	41.389	2.713	23.444
3	**M032[d cm]S[O ApeAprAdT]**	**97.9**	**30.721**	**2.883**	**30.645**
4	M064[d cm]S[O ApeAprAdT]	97.9	32.159	3.077	40.400
5	M128[d cm]S[O ApeAprAdT]	97.7	36.630	3.150	63.260
6	**M[d cm]S[O ApeAprAdT]**	**97.9**	**30.721**	**3.077**	**30.645**
7	M[OrSi]S[O ApeAprAdT]	96.8	46.382	2.679	16.013
8	M[d cm]S[ApeAprAdT]	45.3	298.759	1.999	27.096
9	M[d cm]S[O AprAdT]	90.3	75.955	2.506	29.012
10	**M[d cm]S[O Ape AdT]**	**97.9**	**33.594**	**2.575**	**26.721**
11	M[d cm]S[O ApeApr T]	97.4	34.255	2.698	28.690
12	M[d cm]S[O ApeAprAd]	97.3	33.095	2.722	28.606
13	M[cm]S[O ApeAprAdT]	96.7	36.610	2.569	27.742
14	M[d m]S[O ApeAprAdT]	97.0	41.403	2.655	30.215
15	M[d c]S[O ApeAprAdT]	97.2	34.945	2.752	28.332
16	M[dtcm]S[ODApeAprAdT]	97.2	35.352	2.740	29.763
17	M[OrSi]S[ODApeAprAdT]	94.8	61.165	2.539	15.829
18	M[dtcm]S[DApeAprAdT]	50.6	270.392	2.007	41.181
19	M[dtcm]S[O ApeAprAdT]	97.5	40.649	2.758	28.917
20	M[dtcm]S[OD AprAdT]	86.4	98.084	2.567	28.735
21	M[dtcm]S[ODApe AdT]	96.8	46.255	2.681	27.909
22	M[dtcm]S[ODApeApr T]	93.9	66.275	2.578	28.057
23	M[dtcm]S[ODApeAprAd]	95.1	55.721	2.609	27.961
24	M[tcm]S[ODApeAprAdT]	95.2	59.602	2.535	27.942
25	**M[d cm]S[ODApeAprAdT]**	**97.6**	**41.915**	**2.912**	**30.325**
26	M[dt m]S[ODApeAprAdT]	96.4	54.344	2.579	28.562
27	M[dtc]S[ODApeAprAdT]	96.8	39.204	2.615	28.272
28	**M[dtcm]S[O]**	**86.0**	**94.781**	**2.628**	**27.442**
29	M[OrSi]S[O]	83.3	96.557	2.522	14.972
30	M[ ]S[O]	82.4	101.750	2.414	14.525

However, the minor decline in completeness is accompanied by a significant increase in the average number of steps needed to solve instances in the dataset. Changing the architecture to a single-head scheme (RFL_1Dns) leads to a further decrease in performance to the completeness of 95.4%. We also evaluated the default universal model RFL_rllib, which performs the worst, achieving significantly lower completeness of 90.6%. The main difference between the architecture of RFL_rllib and the other variants is the absence of a shared dense layer before splitting the outcomes into the value and policy branches. Therefore, we argue that the presence of a shared dense layer is a vital part of the proposed architecture as it carries the most significant difference in completeness. Finally, the results of our ablation study consistently show that layer normalization is an inappropriate architecture component for the given label-related modalities, always resulting in an overall performance drop. In the following text, we simplify our notation and use RFL to denote the RFL_Conv model.

5.4 Observation Ablations

Similarly to the previous section, we ablated the observed modalities to justify our design choices of representing the agent's state. We start with the proposed observation set and then systematically remove/vary the included modalities fed into the best-performing architecture RFL_Conv. As described in Subsection 4.2, the proposed vector of observed modalities o_t consists of mapping modalities denoted by **M** and self-aware modalities denoted by **S**. Ray-based mapping modalities include distance to the nearest intersection **d**, type of the nearest intersected object **t** (i.e., label,

anchor, bounds of the environment), and count **c** and mass **m** of labels that the ray went through. Self-aware observations include overlap **O**, displacement **D**, anchor penetration distance **Ape**, anchor-port distance **Apr**, anchor-origin distance **Ad**, and time step **T** modalities. To investigate the importance of ray-based modalities, we also include mapping modalities based on the agent's origin **Or** and size **Si** as a replacement of ray casting.

The results are presented in Table 2. For clarity, we provide a numerical notation of each set of evaluated modalities (*i.e.*, numbers in the first column of the table). First and foremost, the results show that removing the overlap modality (comparing sets 18 with 16 and 8 with 6) dramatically degrades the performance and highly impacts the average number of steps needed to solve the labeling. Therefore, we argue that the overlap modality is the most crucial modality that effectively allows an agent to avoid label-label conflicts. Similarly, removing the anchor penetration modality Ape (comparing sets 20 with 16 and 9 with 6) leads to a significant performance decline and moderately impacts the average number of steps needed to solve the labeling. We argue that the penetration modality effectively allows an agent to avoid label-anchor conflicts. Replacing the mapping ray-based modalities with modalities based on the agent's origin Or and size Si (comparing sets 17 with 16) advocates the presence of ray casting in our method. Ray-based modalities deliver (a) higher completeness and (b) effectively cut the average number of needed steps nearly to half. Interestingly, further removal of displacement modality D (comparing set 19 with 16) positively impacts the performance. We argue the displacement modality (cumulative distance traveled) decreases the number of steps needed for an expense of lower completeness. A similar explanation applies to ray-type modality (comparing set 25 with 16). Therefore, we omitted displacement and ray-type modality due to the negative impact on completeness and conducted a second round of ablations (set 15 to 6). The results show an additional increase in completeness and a decline in the number of steps needed, please compare set 16 with 6. Further findings remain consistent with the first round of ablations (set 27 to 16) except for omitting the proximity modality (set 10) that achieves the same performance, but a higher number of steps is needed in comparison with set 6.

Finally, we experimented with a number of rays, as shown by sets 1 through 5, where **the number following M** denotes the number of casted rays. The results indicate that the best option is to cast 32 and 64 rays to achieve the best completeness. However, casting 64 rays demands more resources, leading to increased observation creation time compared to 32 rays. Reducing as well as increasing the number of rays leads to lower completeness. We hypothesize that the information is likely too sparse at the lower end, not providing sufficient detail. Conversely, opting for 128 over 64 rays might not offer substantial additional information and, combined with an undersized convolution filter, lead to a slight decrease in performance. Nevertheless, the confirmation of our hypothesis remains open for future research. In the following text, we simplify our notation and use RFL to denote the RFL_Conv model combined with modalities of set 3.

6 Comparison with State-of-the-Art

We employ the completeness metric and benchmark dataset, as outlined in Subsection 5.1 and Subsection 5.2, to compare our Reinforced Labels (RFL) with several published methods, beginning with the implementation of Particle-Based Labeling (PBL) proposed by Luboschik *et al.* [11]. The latter method attempts to position each label sequentially, first with the fixed 4- and 8-position model, then using the slider model, and finally with a spiral-based distant model. Furthermore, the method is greedy, and such cannot change the position of a label after it has been placed. Moreover, labels that the method cannot position without a conflict are removed from the calculated label layout. Therefore, the approach may produce an incomplete label layout. We separate PBL into two variants. The first, denoted as PBL-A, involves only the fixed 4-, 8-position, and slider models resulting in **adjacent** label placement. The other variant, referred to as PBL-AD, applies the spiral-based distant model in addition to the fixed 4-, 8-position, and slider models, resulting in a combination of both **adjacent and distant** label placement (*i.e.*, a label can be placed farther away from its anchor, and a leader line maintains the correspondence). Please be aware that PBL-AD cannot be directly compared with adjacent-only methods, such as RFL, given the fact that the distant labels offer a greater degree of freedom. As a result, we expect PBL-AD to achieve a higher completeness score than adjacent-only methods. Nevertheless, we include the PBL-AD in the evaluation to compare the RFL with a method that uses distant labels.

Additionally, we compare RFL to the Rapid Labels (RAPL) [13], a GPU-accelerated greedy and adjacent-only method that leverages the 8-position model. Similarly to PBL, RAPL may produce an incomplete layout as labels that cannot be placed without a conflict by the method are removed from the calculated layout. Table 3 presents a comparison of the evaluated label placement methods.

Finally, we introduce an untrained version of RFL with randomly initialized weights (abbreviated as RFL-random) to validate that the RFL learns a reasonable policy. Comparison with the other existing methods, as discussed in Section 2, is infeasible due to the unpublished or proprietary implementations, which we were unable to get or re-implement.

6.1 Quantitative Results

We trained the RFL policy on randomly generated instances previously described in Subsection 4.1 with at the most two anchors, therefore with up to two agents within the environment, for an hour on a computation node equipped with 2x AMD EPYC™ 7H12, 64-core, 2.6 GHz CPUs without a GPU accelerator. To optimize the parameters of the neural network, we utilize PPO implementation from the RLLib framework [51]. Specifically, we used 119 cores for rollout workers to collect agents' experiences (*i.e.*, observations and rewards) and a single core for the trainer worker responsible for updating the parameters of the proposed network. A detailed description of our training setup is available in the supplementary material. We executed the following evaluations on Intel® Core i7-9700K 8-core, 3.60GHz CPU, and NVIDIA GeForce GTX 1660 Ti. To capture the stochastic nature of RFL (see the definition of Equation 3), we evaluated the RFL and RFL-random ten times over the benchmark dataset. The other compared methods, PBL-A, PBL-AD, and RAPL, are deterministic, and as such, we evaluated each only once. Therefore, we provide quartiles Q1 and Q2, mean and median statistics for the completeness metric of RFL. Furthermore, we fixate the episode horizon of RFL and RFL-random in the evaluation phase at $T = 500$ steps (recall we set the $T = 100$ for the training).

Figure 5(a) and Figure 5(d) show the completeness of the compared methods on the compact and volume datasets. Both charts reveal a similar trend. As the number of anchors rises, as does the environment *occupancy* (*i.e.*, a ratio of the total area of

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729

TABLE 3
Comparison of the evaluated label placement methods. The desirable properties are depicted in green, and less desirable properties are marked in orange and red. The computation time requirements are contingent upon the specific use case. For scenarios where labeling needs to be pre-computed with a high level of completeness, such as cartographic maps, technical drawings, or medical atlases, the RFL method is an appropriate choice. Conversely, in interactive applications such as games or viewers, the trade-off of decreased labeling completeness may be acceptable to ensure faster computation times. Please note that the *Position model* for the PBL-AD and PBL-A methods is *Fixed/Slider* as these methods first try to position labels on the fixed positions, and only if that is not possible, they try several positions between the fixed ones.

	RFL	**RAPL**	**PBL-AD**	**PBL-A**
Paradigm	Machine Learning	Algorithm	Algorithm	Algorithm
Position model	Slider	Fixed	Fixed/Slider	Fixed/Slider
Problem definition	Reward	Value based on rules	Rules	Rules
Label position	Continuous	Discrete	Discrete	Discrete
Heuristic	Non-greedy	Greedy	Greedy	Greedy
Computation time	Non-interactive	Highly interactive	Interactive (when using collision particles)	Interactive (when using collision particles)
Completeness	High	Low	High (achieved by distant labels)	Low
Resolution-independent	Yes	No	Yes	Yes

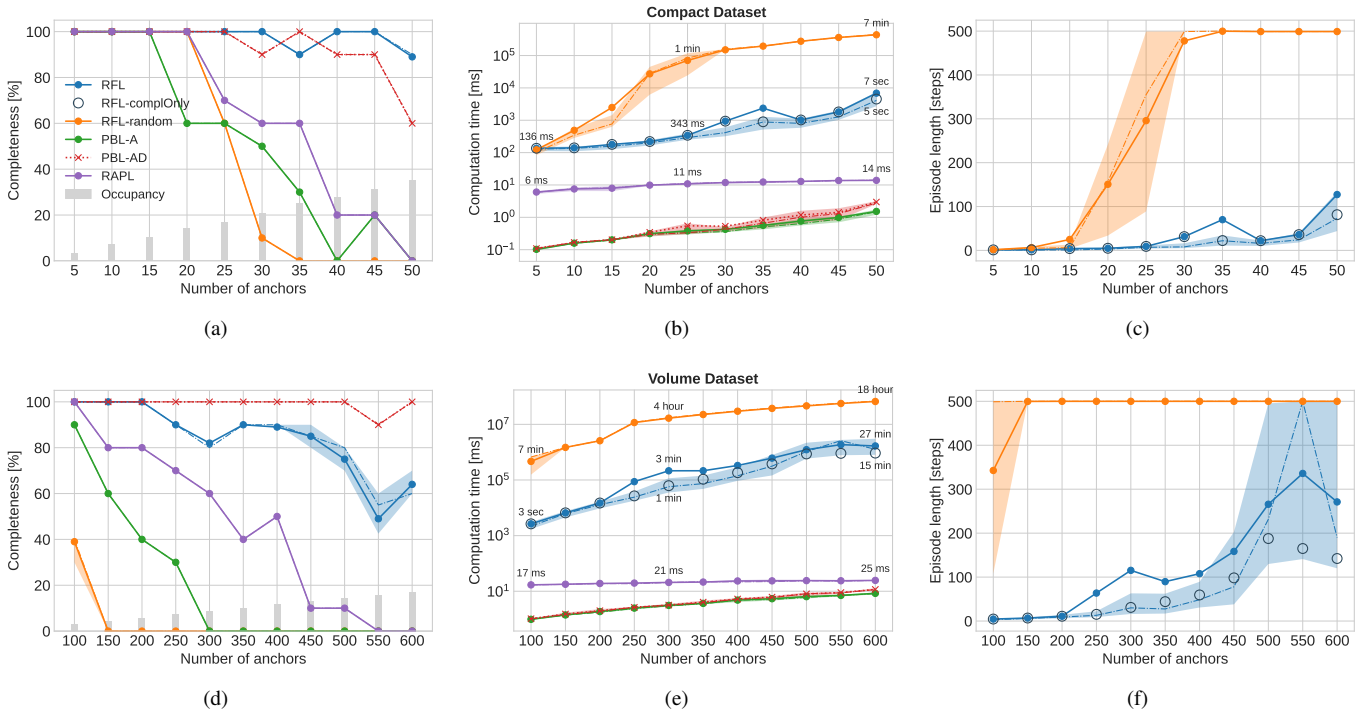


Fig. 5. Comparison of the examined methods evaluated on the benchmark dataset. Charts (a) and (d) illustrate the completeness on compact and volume datasets. Charts (b) and (e) show computation time. Charts (c) and (f) depict the length of the episode. We recall that PBL-AD combining both **adjacent and distant** models cannot be directly compared with **adjacent**-only methods such as RFL, PBL-A, or RAPL.The former PBL-AD provides a greater degree of freedom by placing a label further away from its anchor while the additional leader line maintains the correspondence. We acquired the values for RFL and RFL-random by executing these methods ten times over a set of instances with the same number of anchors. The solid line represents the mean, the shaded area depicts quartiles Q1 and Q2, and the dash-dot line describes the median. The empty circle symbol represents the mean value of complete-only instances found by RFL (incomplete instances reaching the fixed horizon are filtered out). In simpler terms, the value conveyed by the symbol provides an answer to the question: "If and only if the RFL can solve the given instances, how long does it take on average?"

the labels to the overall area of the drawing), the completeness of PBL-A, RAPL, and RFL-random decreases rapidly. The reason behind the difference in completeness between 50 anchors in the compact and 100 anchors in the volume dataset is the varying occupancy. It is easier to find complete labeling within a larger space of volume dataset. Besides PBL-AD, which has the advantage of distant labels, the RFL achieves the highest average completeness of 89% and 64% for 50 and 600 anchors, respectively. The random policy of an untrained agent RFL-random corresponding with a chance performs the worst. This fact confirms that the RFL learns a meaningful policy, and its performance is not the outcome of the randomized search. The second worst method in terms of completeness is PBL-A, followed by RAPL. Starting from 300 anchors, PBL-A fails to place all the labels for any instance, resulting in 0% completeness. Similarly, at 50 and 550 anchors, PBL-A, RAPL, and RFL-random all achieve 0% completeness.

We also compared RFL with PBL-AD, which combines **adjacent and distant** labels and, therefore, has a higher degree of freedom than the strict slider model of our method. However, it is essential to recognize that the inherent flexibility of PBL-AD, stemming from its ability to position distant labels away from the anchors and the use of leader lines for connection, makes it directly incomparable with adjacent-only methods such as RFL. Moreover, the leader lines that grant PBL-AD its flexibility can also intersect with other labels,

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729

anchors, or leader lines, creating potential conflicts. Such conflicts not only diminish the labels' readability but also complicate the association between labels and anchors. Imhof [50] underscores the importance of both label readability and straightforward label-to-anchor association in label layout. Hence, while PBL-AD might seem advantageous regarding placement flexibility, it should be approached cautiously, especially when label layout quality is essential. To our surprise, the PBL-AD performs slightly worse on the compact dataset than RFL and surpasses RFL only at the point of 35 anchors. We further investigated this case and found that the dip in RFL performance is caused by the factual inexistence of a complete conflict-free layout. Therefore, the proposed method outperforms both hand-crafted algorithms RAPL and PBL-A/-AD on the compact dataset. On the volume dataset, the PBL-AD shows stable completeness of 100% through the dataset, except at 550 anchors, where the completeness dips slightly to 90%. We attribute the superior performance of PBL-AD on the volume dataset to the fact that anchors are not spread as evenly across the entire space as in the compact dataset. As a result, PBL-AD can utilize more distant labels for anchors in dense clusters and position them in less dense areas.

Figure 5(b) and Figure 5(e) depict the dependence of the computation time on the number of anchors. We group all instances with a given number of anchors and compute the aggregated statistics. Therefore, we report quartiles Q1 and Q2, mean and median statistics. The PBL-A and PBL-AD are the fastest methods over the entire benchmark dataset. The RAPL follows with a difference of an order of magnitude that steadily decreases towards 600 anchors. This fact goes along with the authors' statement that the performance gain comes with a more significant number of anchors due to the computation of the Summed Area Table [13]. The second slowest method is RFL. We attribute this to the RLLib's internal inefficiencies (*i.e.*, policies among agents within an environment cannot be evaluated in parallel) and the utilized single-thread ray casting implementation from the Box2D framework[3]. In fact, the observation computation and collection take on average 46% (29 ms) of the computation time per step, of which 2/3 makes up the ray-casting operation. The inference of the proposed architecture carries only about 7% (4 ms) of the time. The final 47% (30 ms) is dissolved in preprocessing observations and actions within the RLLib framework. The slowest method overall was RFL-random, hitting the upper bound of 500 steps. Again, this fact confirms that RFL learns a reasonable policy, and its performance is not an outcome of the randomized search.

Figure 5(c) and Figure 5(f) illustrate the dependence of RFL and RFL-random on episode length and the number of anchors. The results significantly distinguish the policy of trained and untrained agents corresponding with a chance that a random set of actions reach the complete conflict-free labeling. The difference is mainly visible at 30 and 150 anchors – the policy of RFL-random skyrockets to the horizon of 500 steps while RFL tops 32 and 7 steps at the same point, respectively.

6.2 User Study

Along with the quantitative evaluation, we conducted a user study to determine the preferred methods among users. We assessed the RFL method, as well as the state-of-the-art methods RAPL, PBL-A, and PBL-AD, which were also evaluated in the quantitative analysis. To carry out the user study, we utilized instances from the compact dataset containing 30, 40, and 50 anchors and labeled each instance with all the evaluated methods. The labeled instances can be found in the supplementary material. We opted for these instances to assess each method with increasing occupancy while avoiding overwhelming the participants when comparing the layouts.

3. The framework Box2D is available at https://box2d.org.

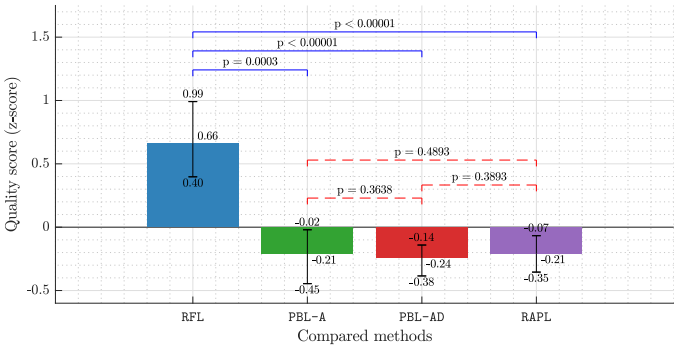


Fig. 6. Results of the user study. The chart shows quality scores accompanied by 95% confidence intervals. Statistically significant differences between the pairs of methods are denoted by solid blue brackets, with the corresponding p-values reported above them. Conversely, the red dashed brackets represent the pairs of methods without evidence of statistically significant differences.

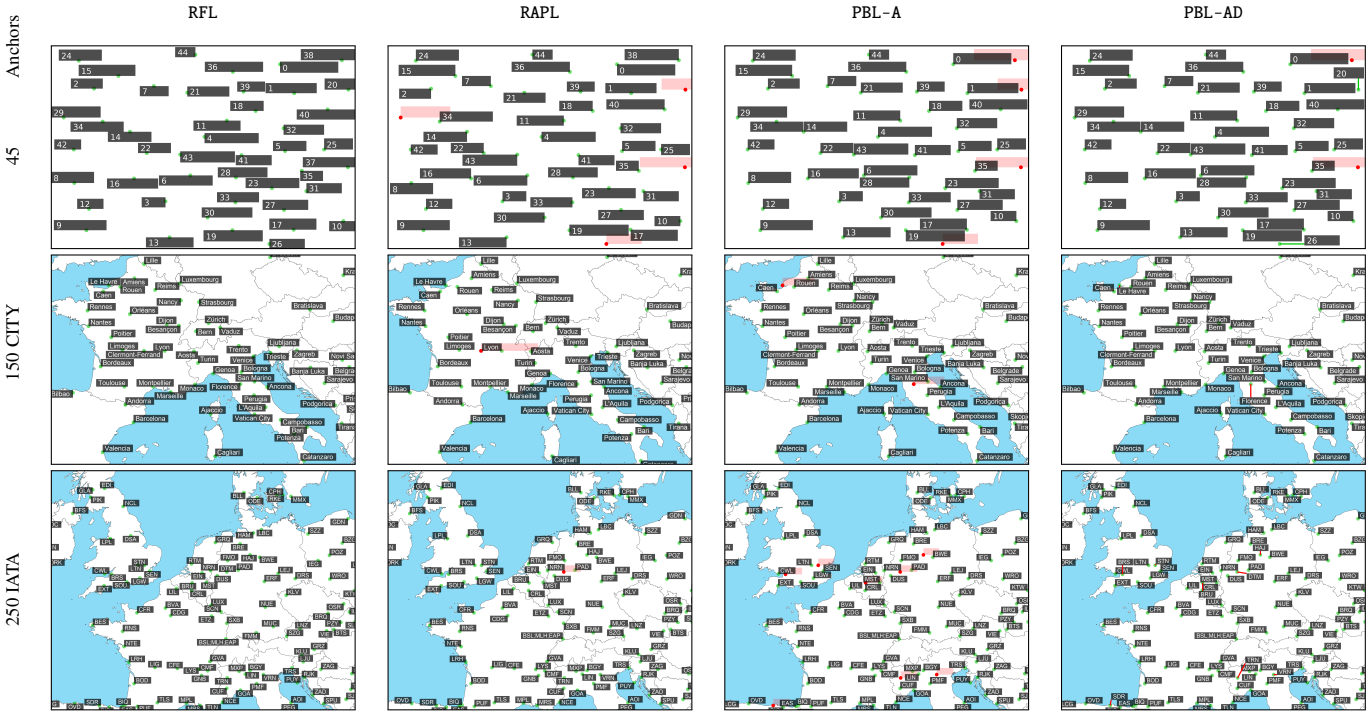
We have designed the user study based on the psychophysical technique of paired comparisons [52]. Specifically, we utilized the two-alternative forced choice (2AFC) paradigm. Each participant was sequentially presented with all possible label layout pairs where, in each pair, the label layouts of the same instance were used and created with different methods. The participants' task was to choose their preferred label layout for each pair in the sequence. To mitigate the learning effect and fatigue, we randomized both the order of the pairs in the sequence and the positions of label layouts (left or right) in pairs. The user study was conducted with 21 participants, consisting of 19 males and two females, with an average age of 21.53 years (ranging from 21 to 24). The average experiment completion time was 6 minutes and 44 seconds, with participants taking anywhere from 2 minutes and 54 seconds to 12 minutes and 32 seconds. Two of the 21 participants were removed using the outlier analysis tool from Pérez-Ortiz and Mantiuk [53], as their results deviated significantly from the others.

We stored the choices in the count matrix $\mathbf{C}$ for each participant individually. Each element c_{ij} in the matrix indicates the number of times that method i was selected over method j. We transformed the per-participant-count matrices $\mathbf{C}$ into a quality score (z-score) scale and calculated statistical significance using a customized MATLAB framework [53]. To transform the matrix $\mathbf{C}$ to the quality score scale, we used Thurstone's Law of Comparative Judgment model concerning Case V [52], [53]. We employed a Two-tailed test with a significance level of $\alpha = 0.05$ to reject the null hypothesis, "there is no clear user preference among the tested methods."

Figure 6 presents the quality scores and the statistical significance of the evaluated methods. The results show that the null hypothesis is clearly rejected as the proposed RFL method exhibits the best quality score that is significantly better than the quality scores of the remaining evaluated methods. In other words, the proposed RFL method was preferred by the users over the remaining evaluated methods. The results also suggest that there is not a significant difference in user preference for the remaining RAPL, PBL-A, and PBL-AD methods.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3313729

TABLE 4
Visual comparison of the examined methods for selected instances from the proposed dataset. In addition, we provide real-world instances of IATA airport codes with 250 anchors and CITY names with 150 anchors based on data obtained from Open Street Maps. For the latter two, we cropped the result to focus on differences among methods. Original results can be found in the supplementary material. The green dot represents the anchor (*i.e.*, point feature). The gray rectangle symbolizes the body of the label itself. The red dot describes an anchor that was not labeled by the given method. The red rectangle illustrates the dimensions of the missing label. We stress that we intentionally added all the missing labels to the visualization for illustrative purposes only, and their origins are not the outcome of the method itself.



6.3 Discussion

The outcome of the comparison is manifold. First, we show that our RL-based method achieves an impressive level of generalization. We remind that we trained RFL on random instances of just two anchors and evaluated the method on unseen instances with up to 600 anchors. The scalability of our method to handle hundreds of agents relies on two key aspects: the design of our environment and the use of Multi-Agent Deep Reinforcement Learning (MADRL). As the number of agents increases, more potential conflicts can occur. However, our local-global reward structure motivates the agents to minimize these conflicts collectively. Notably, our design does not involve any explicit communication channel among the agents. Instead, a shared policy implicitly encourages collaborative behavior, resulting in predictably coordinated actions among agents. Even so, RFL outperformed the compared methods in the category of completeness, and at the same time, the user study participants preferred RFL over all compared methods. By this fact, we demonstrate the power of machine learning techniques and their capability to surpass the hand-crafted algorithms.

Second, we show that greedy methods frequently produce suboptimal solutions concerning completeness. The quantitative results shown in Subsection 6.1 and the comparison of the examined methods in Table 4 provide evidence supporting this claim. For example, in instance 45, RAPL and PBL-A left four anchors unlabeled. Even PBL-AD, with the benefit of distant labels, did not find complete labeling and left two anchors unlabeled. In contrast, RFL produced complete adjacent labelings for all these instances.

Third, we observe a trade-off between optimality and computation time demands. All the previous methods we examined, PBL-A, PBL-AD, and RAPL, can be computed faster but at the expense of an incomplete solution. In contrast, the proposed RFL method is several orders of magnitude slower but, on the other hand, provides results with a much higher level of completeness. Therefore, the examined methods, PBL-A, PBL-AD, and RAPL, are suitable for interactive applications where incompleteness is not critical (interactive visualizations with the ability to zoom). On the other hand, RFL is better suited for cases where the labeling can be computed in advance, and completeness is essential (*e.g.*, cartographic maps, technical drawings, medical atlases). We argue that our approach can serve better than the other examined methods to aid professional illustrators. We believe that future research based on RFL can further mitigate the gap between optimality and speed.

Finally, even though we have primarily focused on the elimination of label overlap in this work, the RL framework is much more versatile. It enables the integration of other metrics into the reward function, allowing one to tailor the solution to specific tasks and opening up numerous opportunities for further improvements and research. The flexibility solidifies the potential of RL in solving complex spatial decision-making tasks like label placement, promising exciting advancements in the visualization field.

7 Limitations

We are aware of several flaws and limitations of the proposed method. First and foremost, RFL is currently limited to finding only

binary solutions. As a result, the method can find either complete conflict-free labeling or a complete but conflict-present solution (*i.e.*, label-label or label-anchor at the screen bound). However, the flexible design of the `AdjacentPFLEnv` environment allows one to define new actions or entirely redefine the existing ones. In future work, adding a further indicator action, which decide whether to place the label or not, could address the binary limitation of the current method. Similarly, the reward objectives could be extended to minimize the number of unlabeled anchors. To this end, we intend to make the `AdjacentPFLEnv` publicly available to further support the research in this domain.

Computation time is another area for improvement of the proposed method. We showed in Subsection 6.1 that `RFL` is magnitudes of order slower than the compared methods. Further examination revealed that the computation and collection of observations contribute significantly to the computation time, mainly due to expensive ray casting and imperfect code optimization. However, we believe that in future work, the limitation can be addressed, for instance, by using a graph representation of possible conflicts known as a conflict graph in the observation. Nevertheless, we leave the question open for future research.

8 Conclusion

In this work, we introduced the first Multi-Agent Deep Reinforcement Learning formulation of the adjacent-point-feature labeling problem. To facilitate the label placement policy training, we developed `AdjacentPFLEnv`, an environment where agents collect experiences – sense the state of the environment via proposed observation modalities, perform actions, and receive feedback in the form of proposed reward. Furthermore, we designed an efficient yet straightforward feedforward neural network architecture with less than half of a million parameters to model the agent's policy and estimate the value function. We show that our approach significantly outperforms previous hand-crafted methods designed by human experts in the number of placed labels and perceived quality. Additionally, we would like to encourage the labeling community towards standardized evaluation, a long-used machine learning practice. To this end, we are proposing a new benchmark dataset to facilitate the comparison of label placement methods, as most of the method codes remain unpublished or proprietary.

Acknowledgments

This work was supported by project *LTAIZ19004 Deep-Learning Approach to Topographical Image Analysis*; by the Ministry of Education, Youth and Sports of the Czech Republic within the activity INTER-EXCELENCE (LT), subactivity INTER-ACTION (LTA), ID: SMSM2019LTAIZ and by Grant Agency of CTU in Prague grant No. SGS22/173/OHK3/3T/13 - Research of Modern Computer Graphics Methods 2022-2024. Computational resources were mainly supplied by the project *"e-Infrastruktura CZ" (e-INFRA CZ ID:90140)* supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] P. Yoeli, "The Logic of Automated Map Lettering," *The Cartographic Journal*, vol. 9, no. 2, pp. 99–108, 1972.

[2] J. Marks and S. M. Shieber, "The computational complexity of cartographic label placement," Center for Research in Computing Technology, Harvard University, Tech. Rep., 1991.

[3] B. Chazelle and A. R. Forrest, "Application challenges to computational geometry: CG impact task force report," in *Advances in Discrete and Computational Geometry*. Providence, RI: American Mathematical Society, 1999, pp. 407–463.

[4] S. Zoraster, "Integer programming applied to the map label placement problem," *Cartographica*, vol. 23, pp. 16–27, 1986.

[5] J. Christensen and J. Marks, "An Empirical Study of Algorithms for Point-Feature Label Placement," *ACM Transactions on Graphics (TOG)*, vol. 14, no. 3, pp. 203–232, 1995.

[6] M. Yamamoto, G. Camara, and L. A. Nogueira Lorena, "Tabu search heuristic for point-feature cartographic label placement," *GeoInformatica*, vol. 6, no. 1, pp. 77–90, 2002.

[7] A. C. Alvim and É. D. Taillard, "POPMUSIC for the point feature label placement problem," *European Journal of Operational Research*, vol. 192, no. 2, pp. 396–413, 2009.

[8] F. Lu, J. Deng, S. Li, and H. Deng, "A hybrid of differential evolution and genetic algorithm for the multiple geographical feature label placement problem," *ISPRS International Journal of Geo-Information*, vol. 8, no. 5, 2019.

[9] S. A. Hirsch, "An Algorithm for Automatic Name Placement Around Point Data," *The American Cartographer*, vol. 9, no. 1, pp. 5–17, 1982.

[10] K. Mote, "Fast point-feature label placement for dynamic visualizations," *Information Visualization*, vol. 6, no. 4, pp. 249–260, 2007.

[11] M. Luboschik, H. Schumann, and H. Cords, "Particle-based labeling: Fast point-feature labeling without obscuring other visual features," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1237–1244, 2008.

[12] C. Kittivorawong, D. Moritz, K. Wongsuphasawat, and J. Heer, "Fast and flexible overlap detection for chart labeling with occupancy bitmap," in *2020 IEEE Visualization Conference (VIS)*, 2020, pp. 101–105.

[13] V. Pavlovec and L. Čmolík, "Rapid Labels: Point-Feature Labeling on GPU," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 604–613, 2022.

[14] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, p. 1334–1373, 2016.

[15] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Large-Scale Data Collection," in *2016 International Symposium on Experimental Robotics*, vol. 1. Cham: Springer International Publishing, 2017.

[16] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," in *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2017, pp. 11.1–11.13.

[17] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[18] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2017.

[19] K. Li and J. Malik, "Learning to optimize," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017, pp. 1–13.

[20] S. Zoraster, "Practical results using simulated annealing for point feature label placement," *Cartography and Geographic Information Systems*, vol. 24, no. 4, pp. 228–238, 1997.

[21] F. Wagner, A. Wolff, V. Kapoor, and T. Strijk, "Three rules suffice for good label placement," *Algorithmica*, no. 30, pp. 334–349, 2001.

[22] R. L. Rabello, G. R. Mauri, G. M. Ribeiro, and L. A. N. Lorena, "A Clustering Search metaheuristic for the Point-Feature Cartographic Label Placement Problem," *European Journal of Operational Research*, vol. 234, no. 3, pp. 802–808, 2014.

[23] M. Yamamoto, G. Camara, and L. A. N. Lorena, "Fast point-feature label placement algorithm for real time screen maps." in *INPE, VII Simpósio Brasileiro de Geoinformática*, 2005, p. 122–135.

[24] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. E. Moret, and B. Zhu, "Map labeling and its generalizations," in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '97. USA: Society for Industrial and Applied Mathematics, 1997, p. 148–157.

[25] M. Kreveld, T. Strijk, and A. Wolff, "Point labeling with sliding labels," *Computational Geometry*, vol. 13, no. 1, pp. 21–47, 1999.

[26] L. Li, H. Zhang, H. Zhu, X. Kuai, and W. Hu, "A labeling model based on the region of movability for point-feature label placement," *ISPRS International Journal of Geo-Information*, vol. 5, no. 9, 2016.

[27] Z. Bylinskii, N. W. Kim, P. O'Donovan, S. Alsheikh, S. Madan, H. Pfister, F. Durand, B. Russell, and A. Hertzmann, "Learning visual importance for graphic designs and data visualizations," in *UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, 2017, pp. 57–69.

[28] Z. Chen, Y. Wang, Q. Wang, Y. Wang, and H. Qu, "Towards automated infographic design: Deep learning-based auto-extraction of extensible timeline," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, pp. 917–926, 1 2020.

[29] 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 Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 294–303, 2021.

[30] 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 *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2389–2399.

[31] R. Hu, B. Chen, J. Xu, O. van Kaick, O. Deussen, and H. Huang, "Shape-driven coordinate ordering for star glyph sets via reinforcement learning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, pp. 3034–3047, 3 2021.

[32] 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 Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 464–474, 2021.

[33] S. Da Col, R. Ciucanu, M. Soare, N. Bouarour, and S. Amer-Yahia, "Dashbot: An ML-guided dashboard generation system," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 4696–4700.

[34] Q. Wang, Z. Chen, Y. Wang, and H. Qu, "A Survey on ML4VIS: Applying Machine Learning Advances to Data Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 5134–5153, 2022.

[35] J. Wang, L. Gou, H.-W. Shen, and H. Yang, "Dqnviz: A visual analytics approach to understand deep q-networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 288–298, 2018.

[36] T. Jaunet, R. Vuillemot, and C. Wolf, "Drlviz: Understanding decisions and memory in deep reinforcement learning," *Computer Graphics Forum*, vol. 39, no. 3, pp. 49–61, 2020.

[37] A. Mishra, U. Soni, J. Huang, and C. Bryan, "Why? why not? when? visual explanations of agent behaviour in reinforcement learning," in *2022 IEEE 15th Pacific Visualization Symposium (PacificVis)*. Los Alamitos, CA, USA: IEEE Computer Society, apr 2022, pp. 111–120.

[38] J. Wang, W. Zhang, H. Yang, C.-C. M. Yeh, and L. Wang, "Visual analytics for rnn-based deep reinforcement learning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 4141–4155, 2022.

[39] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6382–6393.

[40] L. Canese, G. C. Cardarilli, L. D. Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences (Switzerland)*, vol. 11, 2021.

[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[42] M. A. Bekos, B. Niedermann, and M. Nöllenburg, "External labeling techniques: A taxonomy and survey," *Computer Graphics Forum*, vol. 38, no. 3, pp. 833–860, 2019.

[43] P. Bobák, L. Čmolík, and M. Čadík, "Temporally stable boundary labeling for interactive and non-interactive dynamic scenes," *Computers and Graphics (Pergamon)*, vol. 91, pp. 265–278, 2020.

[44] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016. [Online]. Available: https://arxiv.org/abs/1606.01540

[45] D. T. Nguyen, A. Kumar, and H. C. Lau, "Credit assignment for collective multiagent rl with global rewards," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 8113–8124.

[46] S. Schubert, P. Neubert, J. Pöschmann, and P. Protzel, "Circular convolutional neural networks for panoramic images and laser data," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 653–660.

[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[48] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, pp. 26–38, 2017.

[49] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[50] E. Imhof, "Positioning names on maps," pp. 128–144, 1975.

[51] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 3059–3068.

[52] K. Tsukida and M. R. Gupta, "How to Analyze Paired Comparison Data," *UWEE Technical Report 206*, 2011.

[53] M. Pérez-Ortiz and R. K. Mantiuk, "A practical guide and software for analysing pairwise comparison experiments," 2017. [Online]. Available: http://arxiv.org/abs/1712.03686

Petr Bobák is a PhD candidate at the Faculty of Information Technology at Brno University of Technology, Czechia. He received his Master's degree in Computer Science from the same institution in 2017. His research interests include information visualization, machine learning, computer vision, and mathematical optimization.

Ladislav Čmolík is an assistant professor at the Department of Computer Graphics and Interaction of the Czech Technical University in Prague, Czechia. He received his PhD from the same institution in 2011. His research interests include illustrative visualization, non-photorealistic rendering, and HCI.

Martin Čadík is an associate professor of computer science at Brno University of Technology, where he heads his Computational Photography group. He received his PhD from the Czech Technical University in Prague in 2002. His research includes high dynamic range imaging, image processing, computer vision, human visual perception and image, and video quality assessment, among others.