The Journal of Supercomputing (2023) 79:6040–6069
https://doi.org/10.1007/s11227-022-04882-w

Deep AI military staff: cooperative battlefield situation awareness for commander's decision making

Chang-Eun Lee[1] · Jaeuk Baek[1] · Jeany Son[2] · Young-Guk Ha[3]

Accepted: 5 October 2022 / Published online: 31 October 2022

Abstract

There are many studies adopting artificial intelligence (AI) to develop core technologies for the future army but they are still at the level of basic research. It is expected that military power will be negatively affected by aging and declining population. In addition, as more than 500,000 agents will be dispatched to monitor combat scenes, the data sensed by each agent should be managed simultaneously recognize and evaluate the situation on the battlefield in real time. Despite increased complexity in the battlefield, current command system entirely rely on the experience and expertise of individual commanders, which severely restricts defense capabilities. Therefore, AI based military staff needs to be developed to identify potential threats that commanders are likely to miss, to develop smart command systems, and to provide data-driven rationale for commander's decisions. In this paper, we propose a deep AI military staff to support commander decision-making. Our proposed model is composed of four key parts: multi-agent based manned-unmanned collaboration architecture (MACA), robust tactical map fusion technology in poor environments (RTMF), hypergraph based representation learning (HRL) and space-time multi layer model for battlefields recognition (STBR). We design an architecture and generate dataset for training the core network. Simulation results are provided to demonstrate the performance of Deep AI military staff.

Keywords Cooperative battlefield situation awareness · Hidden enemy visualization · Fast panoptic segmentation · Hypergraph · Commander's decision making · AI military staff

✉ Young-Guk Ha
ygha@konkuk.ac.kr

Extended author information available on the last page of the article

1 Introduction

In the battlefield environments where global navigation satellite system (GNSS) is not applicable, combatants encounter many buildings and obstacles that are not recognized beforehand. To obtain spatial information, many sensors with diverse capabilities are used by agents, but the data suffers from quality degradation due to irregular and dynamic motions of combatants. Even worse, more than 500,000 agents will be dispatched in the near future, so the data sensed by each agent should be managed simultaneously to recognize the battle situation in real time manner. In such complex battlefield environments, it is necessary to develop manned/unmanned collaboration system for commander's decision making. In this paper, we propose a deep AI military staff for manned/unmanned agent collaboration system. The proposed deep AI military staff can create spatial map from visual and location information between agents and target location. Also, it can analyze threat in operation areas and alleviate cognitive burden of commanders by data-driven autonomous decision. To recognize global situation of the entire combat scenes, we adopt multitask scheduling with re-planning, which verifies whether mission of each agent is successful under the cyclic operation structure.

The proposed deep AI military staff is composed of four key parts: multi-agent based manned-unmanned collaboration architecture (MACA), robust tactical map fusion technology in poor environments (RTMF), hypergraph based representation learning (HRL) and space-time multi layer model for battlefields recognition (STBR). To develop each part of our proposed deep AI military staff, we design an architecture and generate a dataset for training the core network. Simulation results are provided to demonstrate the performance of Deep AI military staff.

2 Background

The intelligent command control system is a key power system that supports the commander's decision-making and overall management. It improves the joint chiefs of staff system, army Corps-level system, and Army division-level system by supporting timely and appropriate decision-making from various surveillance and reconnaissance assets. In 2018, the U.S. Army introduced an Integrated Visual Augmentation System (IVAS) system to enhance soldiers' awareness and signed a contract with Microsoft to supply HoloLens 2 [1]. IVAS aims to provide augmented reality with various information (environmental information, operational overview, surrounding terrain and building structures) necessary for soldiers to train and perform their duties, and through the AI chipset that recognizes soldiers' eyes, hands and voices by 2023.

DARPA is already developing military situational awareness technologies to support decision-making. Collection and Monitoring via Planning for Active Situational Scenarios (COMPASS) and Active Interpretation of Disparate

Alternatives (AIDA) are the most representative decision support programs. COMPASS [2] utilizes state-of-the-art AI technology to support the commander's judgment in consideration of complex and multilayered battlefield situations. In addition, DARPA [3] simultaneously analyzes unstructured data entered from various multimedia sources through the AIDA project, a decision support program using multimodal data, generating various hypotheses about events, situations, and so on to support the commander's decision making.

3 Related works

3.1 Panoptic segmentation

Panoptic Segmentation [4] is an approach that produces both instance segmentation and semantic segmentation simultaneously. However, instance segmentation is much harder than object detection or semantic segmentation since it should capture more precise and detailed structure of the instances. In case of panoptic segmentation, which provides semantic segmentation for background as well as a object detection and segmentation masks for each instance, it become more difficult. A naïve approach [4] for panoptic segmentation is to predict instance segmentation and semantic segmentation results separately and merge those outputs into one final panoptic segmentation result. Recently, most approaches use a single shared Feature Pyramid Network (FPN) [5] as a backbone for feature extraction, and then add two branches on the feature extractor to produce final panoptic segmentation outputs [6–9]. There are two categories for instance segmentation (or detection) modules in panoptic segmentation networks; one is a two-stage approach [7–9] which is motivated by Mask-RCNN [10], and the other is a single-stage approach [6, 11], motivated by SSD [12] or YOLO [13]. In our method, we choose single-stage approach to achieve real-time speed.

3.2 Image completion

Image completion, also known as image inpainting, is one of the traditional tasks in a computer vision society. It is a task of reconstructing missing regions in an image based on the overall scene. Most early approaches focus on finding similar contents from other parts or background in the image, and copy them to fill gaps [14, 15]. However, these approaches fail to recover large holes since they cannot capture global context of whole scenes, and also they cannot reconstruct content not present in the background image. Recently, learning based methods [16–18] are proposed to solve this problem. These method generate contents based on semantic information from a large dataset.

3.3 Multi-modal data and graph neural network

In the future battlefield environment where multiple manned/unmanned agents such as warrior platforms, drones, and robots are expected to sense diverse kinds of data, it is critical to process these multi-modal data (e.g., visual, voice, language, graph,

etc.) for the commander's global battlefield environment awareness. However, many challenges have been observed because of difficulties on dealing with heterogeneous data in a consistent way. Even worse, the needs to classify agents that have observed the same local situations is growing for global battlefield situation awareness, but this is not easy because tremendous time-series data are generated by agents.

Recently, multi-modal data has been processed in a consistent way by forming and analyzing graphs and their components (i.e., nodes and their relations). Generally, it is tricky to infer information from graphs, but recent advances in graph neural networks (GNNs) enables many tasks on graphs, such as unlabeled node classification, edge prediction and graph classification. To fully utilize graph structure in node classification, [19] and [20] propose a random walk that generates list of nodes with similar characteristics, then learn the feature vectors of nodes and edges. [21] adopts a metapath to enforce random work to follow the predefined path, which outperforms the pure random walk approaches. Also, multi-head attention mechanism used in natural language process is leveraged in graph [22]. As for graph classification, [23] proposes hierarchical graph representation that sequentially coarsens graph upto a predefined size, then learns the feature vector of entire graph. Based on Weisfeiler-Lehman (WL) algorithm, [24] classifies the graph in a consistent order by sorting graph nodes with the extracted information.

4 System design and major contributions

Our proposed deep AI military staff is composed of four parts; MACA, RTMF, HRL and STBR as shown in Fig. 1. MACA is the entire architecture including RTMF, HRL, and STBR. RTMF includes the ability to create battlefield scenarios, analyze information collected by individual agents, and generate data necessary for learning. HBR includes the ability to select agents that share the same situation/environment by analyzing the similarity between graphs generated by individual agents and to discriminate and confuse the similarity between nodes in the distributed graph to create a global graph. STBR includes a technology that predicts the battlefield situation/environment in real time by analyzing the correlation of complex knowledge distributed in time and space using a fusion knowledge graph generated by HRL. We first focus on RTMF and HRL and provide their major contributions. Then, details of each part of deep AI military staff are illustrated in Sects. 5-8.

4.1 Enemy detection and hidden enemy visualization

Computer vision is the basis of many real-life applications. In order to deal with variety applications in the real-world, numerous computer vision algorithms, such as image classification [25, 26], visual tracking [27], object detection [28] and semantic segmentation [29], have been developed. Thanks to recent great advance in deep learning, computer vision algorithms have shown rapid improvements in the past decade.

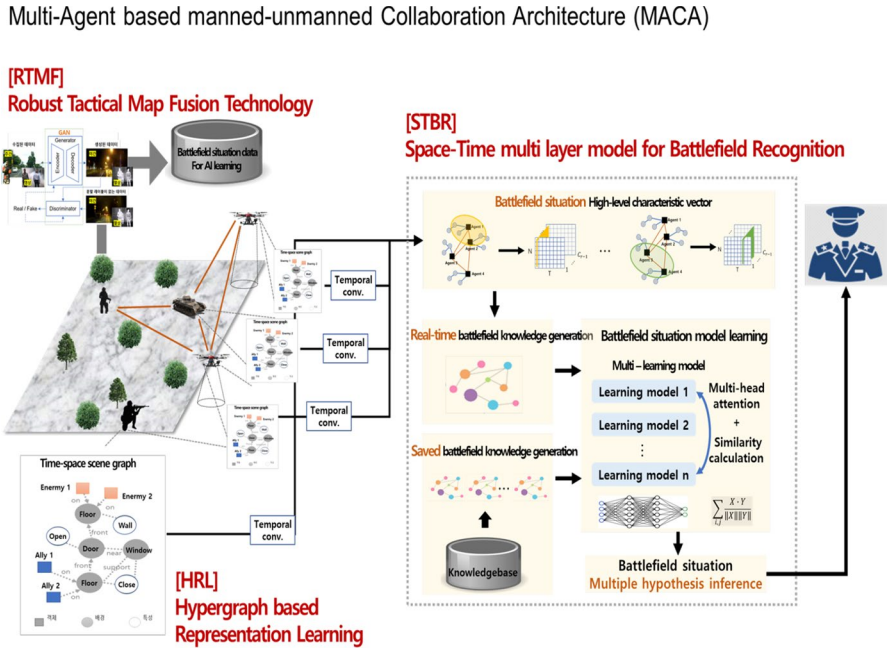

Fig. 1 The architecture of MACA

Those computer vision algorithms are also applied to support visual cognition of soldiers in battlefields. In recent Integrated Visual Augmentation System (IVAS) project, Microsoft's HoloLens2 is used to support displaying important information in battlefields, since it is crucial to understand surrounding scenes for survival of soldiers. However, scene understanding becomes more challenging due to complex, drastic changes in battlefield environments, and target object recognition failures increase. Moreover, since most of enemies hide their bodies by cover and concealment as shown in Fig. 2, it is much hard to detect them in battle. Such frequent encounters with concealed enemies can lead to increased level of fear and tension and drop in the overall moral of the troops.

In Sect. 6, we propose a new method that exposes hidden enemies in complex scenes using panoptic segmentation and image completion methods to overcome these problems. An overall framework of our proposed RTMF is shown in Fig. 3. First, to detect enemies in complex scenes, we design a real-time panoptic segmentation network which runs 33fps on 550x550 resolution without severe performance drops. The feature pyramid network (FPN) [5] is shared for both instance and semantic segmentation to encode shared representation for each work. We also adopt a similar network architecture to YOLACT [30] which is a well-known real-time instance segmentation network, and extend it to panoptic segmentation. Second, to visualize hidden enemies, we utilize Pluralistic Image Completion network (PICNet) [31] to reconstruct occluded body parts.

The main contributions of RTMF in Sec. 6 are listed as follows:

Fig. 2 Invisible enemies due to cover and concealment

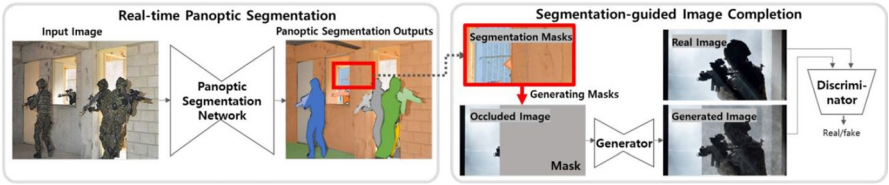

Fig. 3 The overall framework of robust tactical map fusion technology in poor environments (RTMF). A discriminator in the image completion network is only used during training

- We propose a novel framework to reveal and visualize enemies hidden by cover and concealment.
- We implement a real-time panoptic segmentation net-work based on YOLACT to detect enemies and capture characteristics in battlefield scenes.
- We introduce a segmentation-guided image completion method to recover occluded parts of target objects.
- We demonstrate that our segmentation network achieves comparable performance to the state-of-the-art methods and our image completion network is able to recover hidden enemies successfully.

4.2 Hypergraph for similarity analysis

Many studies on GNNs provide successful results on their tasks, but traditional graph structured data still lacks in terms of data representation; since an edge can only connect at most two nodes, it cannot fully represent a real word with many tangled nodes. To tackle these problems, the concept of hypergraph is introduced, where a hypergraph is composed of hypernodes and hyperedges. Specifically, a hypernode represents an entity in a hypergraph and a hyperedge denotes a set of correlated hypernodes without any restriction on their numbers and types. By

constructing a hypergraph, two or more hypernodes with different features can be related one another, allowing us to extract the useful relationship information. Similar to GNNs, deep learning architectures are used in hypergraph for classification [32–34], where the embedding vector of hypernodes is learned by hyperedge convolution operation [32], by processing tuple of multi-modal data [33] and by applying multi-head attention mechanism [34]. However, most of these studies focus on hypernode classification, but do not utilize hypergraph to analyze interrelationship between distributed graphs. Since a lot of agents in battlefield environments measure multi-modal data that can be transformed into graph, it is critical to process distributed graphs by integrating all geographically similar information for global situation awareness.

In Sect. 7, we propose a HRL to learn the embedding vector of distributed local graphs. Based on the fact that multi-modal data sensed by agents can be transformed into graph, our objective is to classify agents that have observed the same local situations by treating each agent as a local graph and training a hypergraph-based deep learning model. After training the embedding vector of agents, we can provide adjacency matrix that shows interrelationship between agents.

The main contributions of HRL in Sect. 7 are as follows.

- Based on local graph from each agent, we construct a hypergraph to integrate multiple graphs and then, utilize a hypergraph random walk to obtain a bunch of training dataset for agent embedding vector.
- Under the predefined probabilistic rules, the proposed hypergraph random walk makes random movement from one hypernode to another, which means *inter-graph* (i.e. graph-to-graph) movements. Note that the conventional random walk in [19–21] makes an *intra-graph* movement from one node to another in the graph.
- Because of unlabeled situation information, we train the agent embedding vector in an unsupervised manner to make the embedding vectors of neighboring agents similar. As a result of similarity analysis, we can provide adjacency matrix of agents, which can be used along with any GNNs to construct global graphs and perform graph convolution for the commander's battlefield situation awareness.

5 MACA: Multi agent-based manned-unmanned collaboration architecture

Figure 4 shows the overall architecture of the proposed system for enhancing awareness of combatants in a building or an underground bunker. The collaborative agent generates a collaboration plan according to a mission, requests neighboring collaborative agents to search for knowledge/devices available for collaboration and review the availability of the knowledge/devices, generates an optimal collaboration combination on the basis of a response to the request to transmit a collaboration request, and upon receiving the collaboration request, performs the mission through mutual distributed knowledge collaboration. Such a collaborative agent may provide information about systems, battlefields, resources, and tactics through a determination

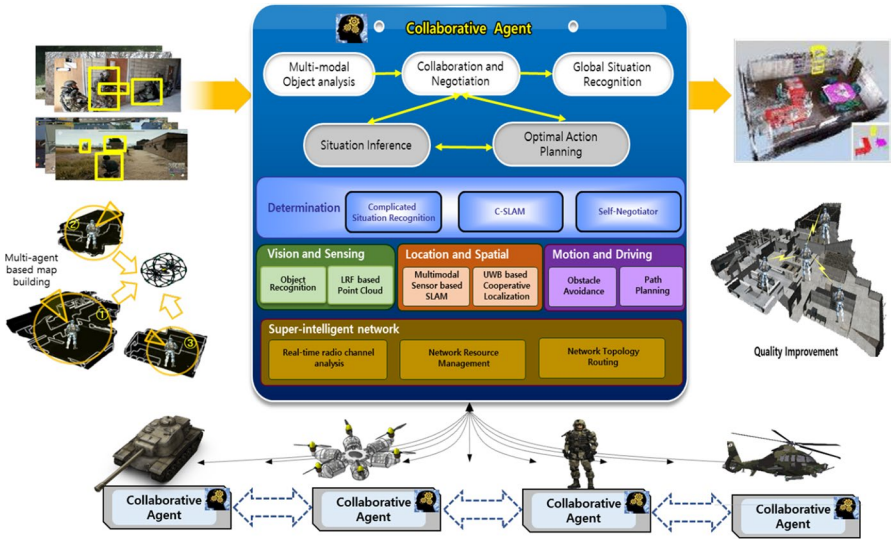

Fig. 4 The proposed multi-agent based manned-unmanned collaboration architecture (MACA)

intelligence processing unit, such as complicated situation recognition, coordinative simultaneous localization and mapping (C-SLAM), and a self-negotiator.

Meanwhile, in order to support a commander in command decision, the collaborative agent combines the collected 5 pieces of information to be subjected to artificial intelligence (AI) deep learning-based global situation recognition and CSLAM technology to provide the commander with command decision information merged with unit spatial maps through the autonomous driving robot linked with the smart helmet worn by the commander.

To this end, referring to Fig. 4, the collaborative agent includes a multi-modal object data analysis unit, an inter-collaborative agent collaboration and negotiation unit, and an autonomous collaboration determination and global situation recognition unit so that the collaborative agent serves as a supervisor of the overall system.

In addition, the inter-collaborative agent collaboration and negotiation unit searches a knowledge map through a resource management and situation inference unit to determine whether a mission model that is mapped to a goal state corresponding to the situation and environment data is present, checks integrity and safety of multiple tasks in the mission, and transmits a multi-task sequence for planning an action plan for the individual tasks to an optimal action planning unit so that the tasks are analyzed and an optimum combination of devices and knowledge to perform the tasks is constructed.

On the other hand, the optimal action planning unit performs refinement/division/allocation on action-task sequences to deliver relevant tasks to the collaborative agents located in a distributed collaboration space on the basis of a generated optimum negotiation result through a knowledge/device search and connection protocol of a hyper-intelligence network formed through the autonomous driving robots so as to deliver the relevant tasks to wearers of the respective smart helmets.

In addition, the autonomous collaboration determination and global situation recognition unit verifies whether an answer for the goal state is satisfactory through global situation recognition monitoring using a delivered multitask planning sequence using a collaborative determination/inference model and, when the answer is unsatisfactory, requests the inter-collaborative agent collaboration and negotiation unit to perform mission re-planning to have a cyclic operation structure.

6 RTMF: Robust tactical map fusion technology

This section describes the details of our proposed frame-work to visualize hidden enemies using panoptic segmentation and Segmentation-guided image completion methods. We first detect enemies by panoptic segmentation, and then using those segmentation results as occlusion masks for the image completion network to recover and visualize hidden parts of an enemy.

6.1 Real-time panoptic segmentation

We design a real-time panoptic segmentation network based on YOLACT [30], which is a well-known single-stage instance segmentation network, to detect enemies. Our panoptic segmentation network consists of three sub-networks: feature pyramid network (FPN), instance segmentation network, and a semantic segmentation network. The final panoptic segmentation outputs are generated by combining two outputs of instance and semantic segmentation networks, as shown in Fig. 5. Our single-stage network has the advantage of a high speed compared to the multi-stage network that uses the Region Proposal Network (RPN) separately. In general, there are two types of classes for panoptic segmentation; one is a Things class and the other is a Stuff class. Our instance segmentation network only produces results for a Things class, and the semantic segmentation network performs for all classes.

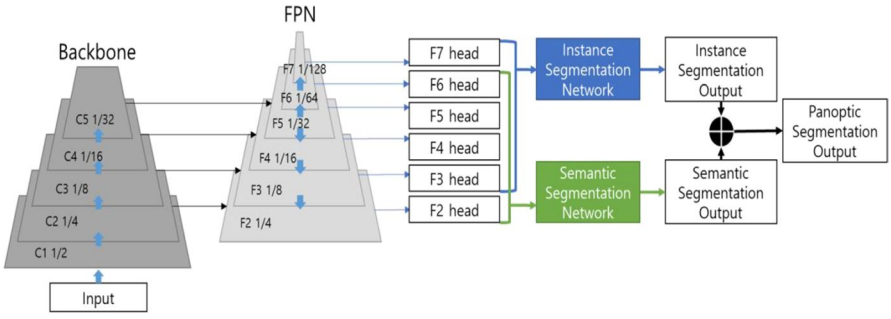

Fig. 5 Overall architecture of the proposed panoptic segmentation network. Our network consists of three sub-networks: feature pyramid network (FPN), instance segmentation network and semantic segmentation network. Final panoptic segmentation outputs are generated by merging two outputs of each segmentation network

6.1.1 Instance segmentation network

We adopt a YOLACT [30] architecture for out instance segmentation network. Most single-stage instance segmentation methods motivated by SSD and YOLO are fast but show worse performance than two-stage ones. To increase performance while keep real-time speed, YOLACT divides an instance segmentation problem into two parallel tasks. One (ProtoNet) is generating a set of prototype masks on images, and the other (Prediction head) is predicting a set of coefficient to compute linear combination with prototype per instance. Two separate networks for these tasks share the FPN [5], and we denote k-level features on FPN by Fk heads for simplicity. Object masks for the anchor boxes of each cell are generated by multiplying the prototype activation by the mask coefficient of the prediction head. Then, the final object mask is obtained by binarization through crop and threshold operations to obtain the final instance segmentation results.

Our loss function for training the instance segmentation network consists of three different losses; a bounding box regression loss, a classification loss, and a mask coefficient loss, as follows:

$$L_{\text{inst}} = L_{\text{boxreg}} + L_{\text{boxcls}} + L_{\text{mask}}, \tag{1}$$

where L_{boxreg} is a smooth-L1 loss to regress offsets for box coordinates of anchors, L_{boxcls} is a softmax cross-entropy loss, and L_{mask} is a pixel-level binary cross-entropy loss between a linear combination of prototype masks and a ground-truth mask for each bounding box. Note that, these losses for the instance segmentation network are only computed for Things classes. For more details of the instance segmentation network, please refer to YOLACT [30].

6.1.2 Semantic segmentation network

To produce segmentation masks for Stuff classes, we combine a semantic segmentation branch into the panoptic segmentation network. Our semantic segmentation network is also attached to FPN and uses feature maps from F2 to F6 heads. To reduce model complexity, the semantic segmentation network employ the shared head used in the prediction head of the instance segmentation network. Thus, the shared head also used to encode semantic features for each features head. This leads to achieving real-time panoptic segmentation by avoiding heavy computation for semantic segmentation. Those features are upsampled and concatenated into one single feature map and feed to one last convolution layer to produce the final semantic segmentation output masks. The detailed structure of our semantic segmentation network is shown in Fig. 6.

To train the semantic segmentation network, we use a cross-entropy loss with pixel-level hard-negative mining [35] where pixels with large losses have more penalty. In our experiments, we penalize on pixels of which losses are larger than 30% percentile of all predictions. Then, the final loss for training the proposed network is as follows:

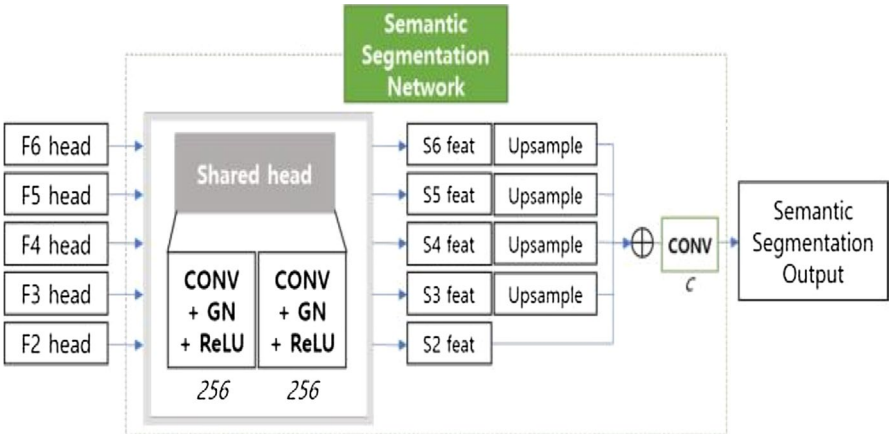

Fig. 6 Detailed architecture of the proposed semantic segmentation network

$$L_{tot} = L_{inst} + \lambda \cdot L_{sem}, \tag{2}$$

where L_{sem} is a pixel-level cross-entropy loss between a segmentation segmentation prediction and a ground-truth segmentation mask generated by panoptic segmentation masks, and λ is a constant and is set to 3 in our experiments.

6.2 Segmentation-guided image completion

In order to visualize hidden enemies, we use an image completion network based on conditional variational auto-encoder with generative adversarial network (cVAE-GAN) which is referred to as pluralistic image completion network (PICNet) [31]. Using the result of the proposed panoptic segmentation as an input mask, enables robust restoration against extreme occlusion. First, we find hidden enemies and their occluded parts using the panoptic segmentation net-work. Predicted panoptic segmentation masks on occluded parts are regarded as a mask to be recovered. This image completion network consists of a generator network that removes the hidden area and a discriminator that determines the image generated by the generator and the original image, as shown in Fig. 7. After learning, only the generator is used to restore the occluded region.Please refer to PICNet [31] for further details.

7 HRL: Hypergraph based agent representation learning

In this section, we propose how to deal with multi-modal data in a consistent way, then construct a hypergraph for inter-graph similarity analysis. By learning the embedding vector of agent (i.e., local graph), we can obtain adjacency matrix of agents that is the key element in GNNs to perform graph convolution for the commander's battlefield situation awareness.

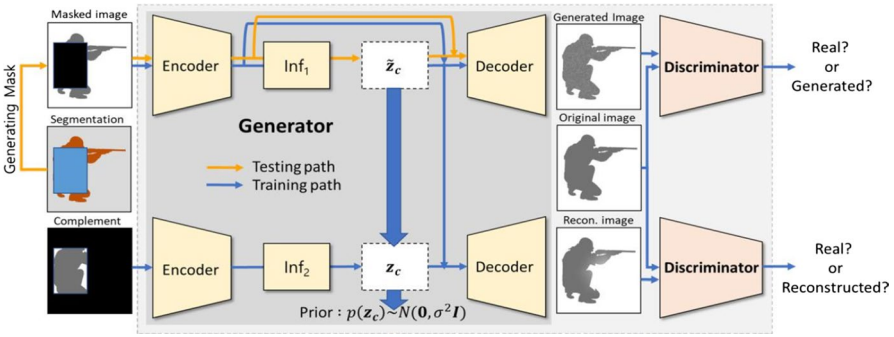

Fig. 7 Overall system of the image completion network based on PICNet [31]. Occluded masks are generated by panoptic segmentation outputs

7.1 Overall procedures

Figure 8 represents system architecture of our proposed hypergraph based representation learning (HRL). Based on multi-modal data sensed by agents, HRL first converts them into graphs. Then, a hypergraph is constructed to integrate multiple graphs, and training data are made by hypergraph random walk. To train an embedding vector of agents, we minimize triplet loss, where anchors, positive samples and negative samples are selected from hypergraph induced information. After training the embedding vector of agents, adjacency matrix is made to represent interrelationship between agents, which can be used in later part of Deep AI Military Staff for the commander's battle field situation awareness. Details of our HRL are illustrated in the following subsections.

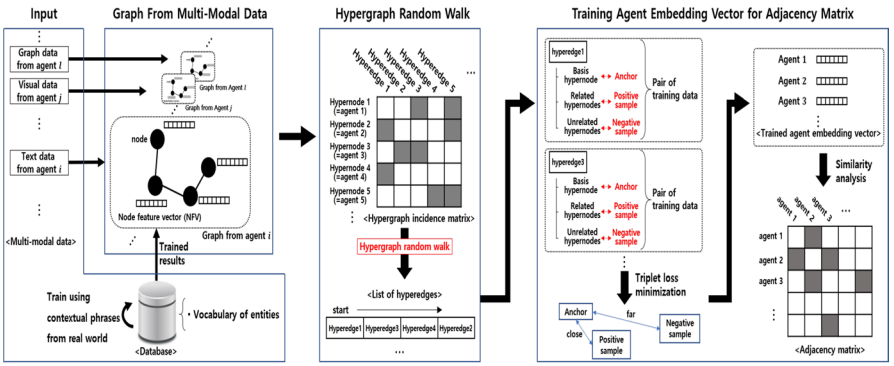

Fig. 8 System architecture for our hypergraph based representation learning (HRL)

7.2 Graph from multi-modal data

In this subsection, we illustrate how to convert multi-modal data into knowledge graphs. A graph is a structure of nodes and edges, where nodes with similar properties are connected by edge to represent their relational information, and a feature vector is assigned to each node to indicate its characteristics. We first describe how to extract graph components from multi-modal data. Then, we explain the needs for a database for node feature vectors, and construct a knowledge graph.

7.2.1 Graph components

From multi-modal data, we can extract entities and attributes that can be reconstructed as contextual phrases. For example, in combat scenes, some snipers with brown hair aim their guns at the enemy while other soldiers with bombs hide behind large trees. Based on an object detection algorithms [13, 28, 37], objects (such as snipers, soldiers, bomb, enemy and trees) can be detected in images and interpreted as nodes in a graph. In addition, visual grounding techniques can be applied to find attributes of each object (e.g., brown hair and large) and relationship between objects, where the former and the latter denote the attributes and edges in graph, respectively. This is an simple example of how to convert image to graph, but other types of raw data (sounds, texts, etc) can also be transformed into graph without any difficulties.

7.2.2 Database

Although nodes, attributes and edges can be obtained directly from raw data, more efforts are required to obtain node feature vector (NFV) because it can be used in any other graphs to indicate unique characteristics of nodes. To this end, we construct database that saves all the entities with their feature vectors. Based on contextual phrases that can be obtained from real world (e.g., combat movies, combat games, or combat histories), we create vocabulary of entities and their feature vectors, where word embedding modules [38] is used to obtain the feature vector of entities (see Fig. 9 for more information). Then, the trained feature vectors of entities are used as a basis to create NFVs.

7.2.3 Knowledge graph

Now we are ready to construct knowledge graph from multi-modal data. Compared to conventional graphs that has nodes, attributes along with adjacency matrix among nodes, we construct knowledge graph with nodes and NFVs, where NFVs are trained to contain information about relations and attributes. Note that in battlefield scenario where multiple distributed graphs need to be considered simultaneously, adjacency matrix of a single graph only carries local structure information and is not meaningful for the entire distributed graphs. So, we train node-relation vector to

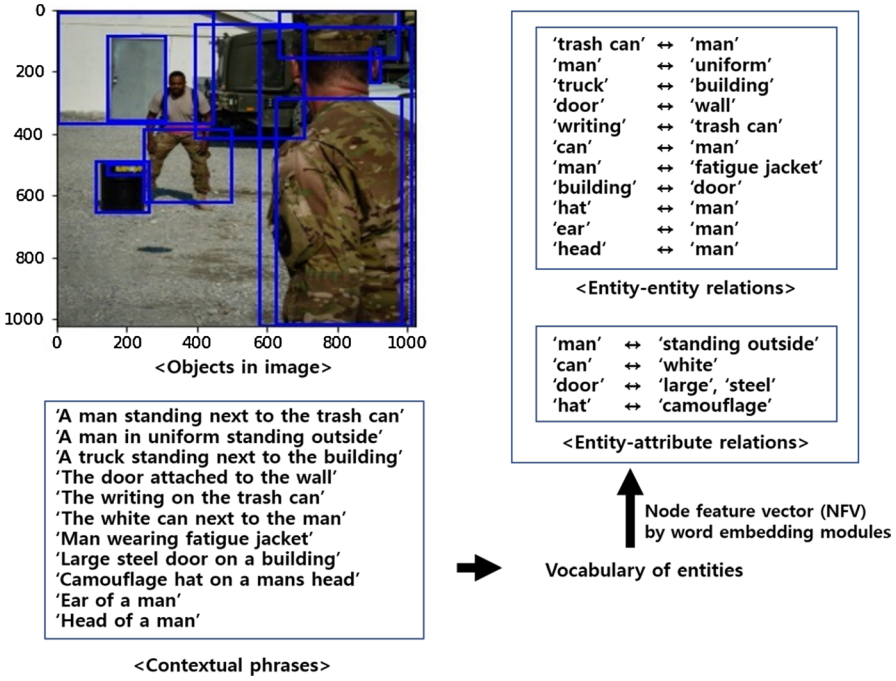

Fig. 9 Graph components and the feature vector of entities. The image and contextual phrases are from visual genome dataset [36]. Many contextual phrases can be made from a single image, and each phrase has entities with attributes, which correspond to the nodes in graph. The feature vectors of entities are trained using word embedding modules and can be used to create NFVs

utilize adjacency matrix implicitly. Also, we train node-attribute vector to represent correlated nodes and attributes, which can treat nodes with and without attributes in a consistent way.

Based on feature vector of entities in database, we train node-relation and node-attribute vectors using [21], and concatenate them to create NFV with fixed size.

7.3 Hypergraph random walk

In this subsections, we construct a hypergraph to integrate distributed graphs and apply hypergraph random walk to obtain training data for agent embedding vector.

7.3.1 Hypergraph

A hypergraph is composed of hypernodes and hyperedges, where a hypernode means an agent, and each node in distributed graphs makes a hyperedge. For example, when each of N agents has a graph with M nodes, a hypergraph is composed of N hypernodes and $N \times M$ hyperedges. Each hyperedge is interpreted as a set of correlated hypernodes and can be defined through similarity analysis

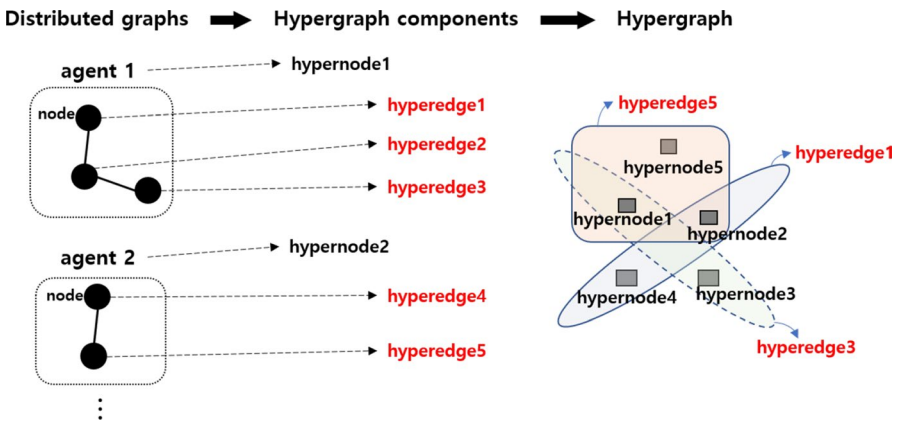

Fig. 10 Example of hypergraph

Hyperdege	Hypernodes	
	Related	Unrelated
1	2, 4	1, 3, 5
2	3	1, 2, 4, 5
3	1, 3	2, 4, 5
4	5	1, 2, 3, 4
5	1, 2, 5	3, 4

Fig. 11 Example of hypergraph induced information

between nodes in distributed graphs. Specifically, by regarding each node as a basis, NFVs from previous subsections are used to sort all nodes by similarity. Then, a threshold or a predetermined number can be used to select the nodes such that agents (i.e., hypernodes) corresponding to the selected nodes would be included in a hyperedge (see an example of hypergraph in Fig. 10).

Once a hypergraph is obtained, a hypergraph incidence matrix can be defined, where a row denotes a hypernode (i.e., an agent) and a column denotes a hyperedge (see the middle part of Fig. 8). The element of hypergraph incidence matrix can be defined as binary, where one is assigned when a hypernode is included in a hyperedge, and zero is assigned otherwise. From the hypergraph incidence matrix, we can extract relationship information among hypernodes. Figure 11 represents an example of hypergraph induced information for the given hypergraph incidence matrix in Fig. 8. It is observed from Fig. 11 that a single hyperdedge relates multiple hypernodes. Also, for each hyperedge, hypernodes can be divided into two sets; one for related hypernodes and others for unrelated hypernodes.

7.3.2 List of hyperedges

Based on hypergraph induced information, hypergraph random walk can be used to obtain training data for agent embedding vector. Since all nodes in distributed graphs have their own information, it is critical to visit all the nodes to carry all information about graphs. Based on that each hyperedge can be defined on each node, our proposed hypergraph random walk defines inter-graphs movement by performing permutation of all hyperedges. Note that one permutation makes the list of hyperedges, and several independent trials of permutations provide a bunch of training data. A batch size for training a network is defined by considering all or part of the hyperedge list.

7.4 Training agent embedding vector for adjacency matrix

In this subsection, we train a network to obtain agent embedding vectors, then construct an adjacency matrix that shows relationship information between agents.

7.4.1 Loss function

The objective of training agent embedding vector is to classify agents that have observed the same local situations. We adopt triplet loss in training because minimization of this loss makes embedding vectors of two samples (i.e., anchor and positive sample) similar, while those of anchor and negative samples are trained to be dissimilar.

Let $\mathbf{\Psi}$ be the set of all hypernodes (i.e., agents), and $\mathbf{\Psi}_P$ and $\mathbf{\Psi}_N$ denote positive and negative samples of hypernodes, respectively. When hypernode $i \in \mathbf{\Psi}$ is used as anchor, triplet loss can be expressed as

$$\boldsymbol{L}(X_i) = \sum_{p \in \mathbf{\Psi}_P} \log \sigma\Big(f(X_i, X_p)\Big) - \sum_{n \in \mathbf{\Psi}_N} \log \sigma\Big(f(X_i, X_n)\Big), \tag{3}$$

where X_i is embedding vector of hypernode $i \in \mathbf{\Psi}$, f is a vector operator for similarity analysis and σ is a Relu function. Note that $\mathbf{\Psi}_P$ and $\mathbf{\Psi}_N$ are updated as anchor changes, and hypergraph induced information (in Fig. 11) can be used to define hypernode pairs of anchor, positive and negative samples. For example, for hyperedge 1 in Figs. 10 and 11, hypernode (i.e., agent) that has graph with node 1 is used as anchor, and related hypernodes (i.e., hypernodes 2 and 4) and unrelated hypernodes (i.e., hypernodes 1, 3, 5) are used as positive and negative samples, respectively.

7.4.2 Adjacency matrix of agents

Once embedding vectors of agents are obtained, we can utilize them to construct adjacency matrix, where each row and column represents the agent. Since each

agent corresponds to graph, adjacency matrix of agents can be used to construct global graph from multiple local graphs.

It is worth noting that adjacency matrix only represents relation between distributed graphs, so it seems that there is no information about which parts of adjacent graphs (i.e., nodes) are connected. However, since hypergraph induced information provides relational information in node level, we can merge multiple local graphs without difficulties.

8 STBR: Space-time multilayer model for battlefields recognition

8.1 Battlefield object model

Figure 12 is a common-based ontological design for distributed agents mounted on situational cognitive systems and individual combatant equipment to identify and share situations in the battlefield environment with the various objects present in the battlefield environment.

The battlefield situation is represented by the spatial interaction of various objects present in the battlefield environment, which requires the representation of each object as a correlated vector in order to learn/recognize effectively based on a deep neural network model. In this paper, for the development of a situation-aware system, we express a formal world model based on the spatial and time-space relationship between the various objects and objects that make up the battlefield environment as shown in Fig. 13.

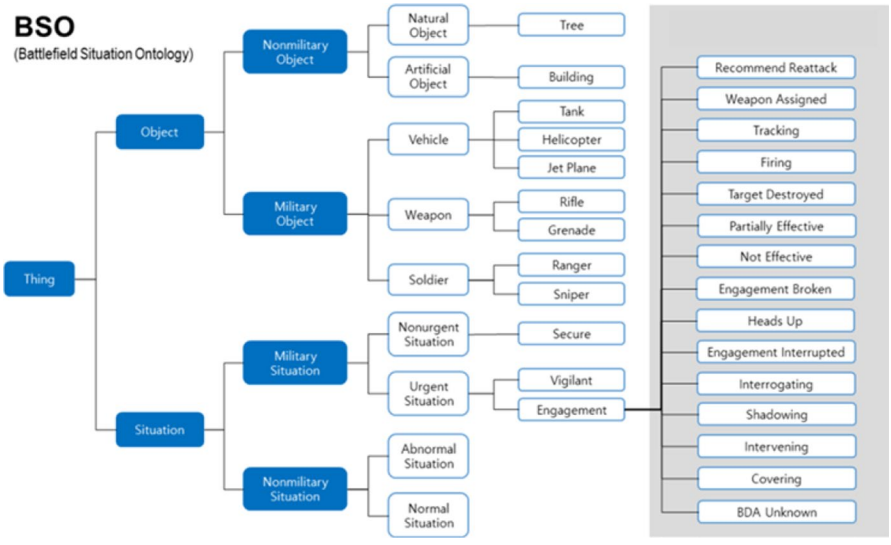

Fig. 12 Battlefield situation ontology (BSO)

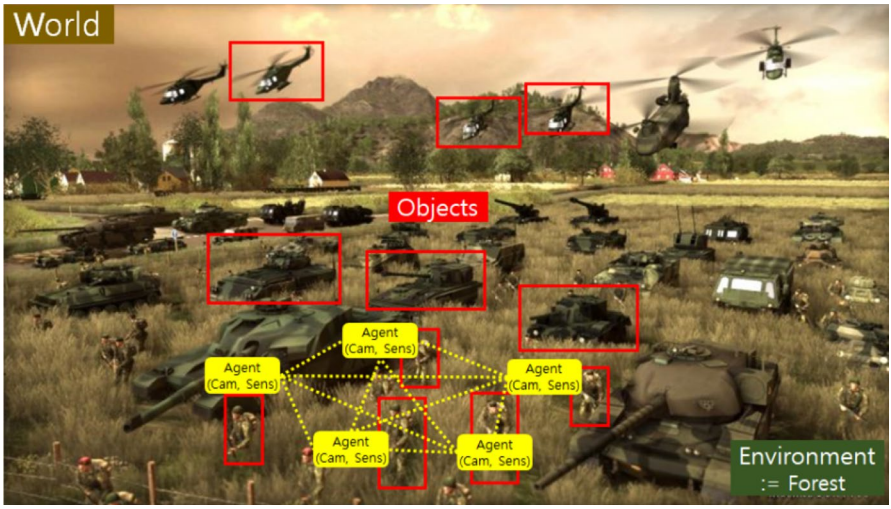

Fig. 13 World model for situation recognition

8.2 Collaborative knowledge based battlefield situation awareness

Existing situational awareness systems use prebuilt situational knowledge-based symbolic reasoning, making it very difficult to infer situational knowledge building or unexpected situations in complex, time-space dynamic environments such as battlefields. in this paper, we present a new technique to continuously learn time-space changes for various situations through mechanical learning based on environmental awareness information collected from the battlefield.

Engagement is a very important situation to be aware of in a battlefield environment, and in order to accurately recognize the situation, it is necessary to recognize light sources such as sparks or explosions from firearms (guns, heavy weapons, etc.). Figure 14 illustrates a machine learning-based process that can recognize a field situation by recognizing the light source in real time based on the texture characteristics of various light sources that can occur in the field.

9 Experiment results

In this section, we evaluate our panoptic segmentation on the public benchmark, and compare our performance to the state-of-the-art methods. Also we show image completion results on the our private dataset constructed using the Battlefield4.

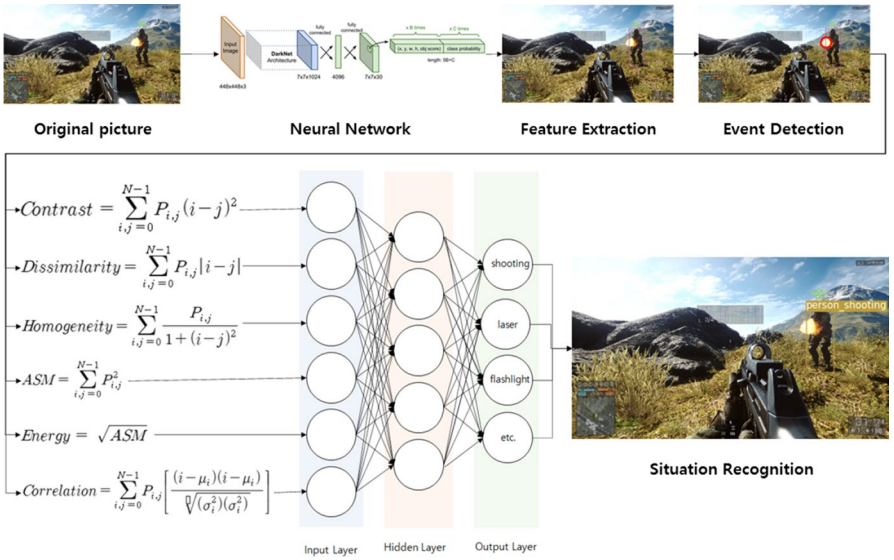

Fig. 14 The machine learning-based process for situation recognition

9.1 Real-time panoptic segmentation in RTMF

9.1.1 Implementation details

We train our model using 3 GPUs with ImageNet pretrained ResNet-50-FPN as our feature backbone, where the batch size is set to 1 per each GPU, and a input image size of 1440 × 720, We freeze BatchNorm layers in the backbone and add GroupNorm [39] layers after backbone. We use Adam Solver [40] for 400k iterations with initial learning rate of 103 and weight decay of 104. Learning rate is decreasing by a factor of 0.1 at 28 and 36k.

9.1.2 Datasets

We evaluate our method on the Cityscapes panoptic segmentation benchmarks [41]. The Cityscapes dataset consists of street scenes with a total of 19 classes, where 8 classes for things which have instance-level labels as well as semantic class labels, and 11 classes for stuff which only have semantic labels. Following the standard protocol, we use 2975 image for training and 500 images for testing with a resolution of 1024 × 2048 for evaluation. Note that we only used fine-grained labels for training. For data augmentation, color jiterring, random crop, random scaling, random flipping are applied. We also test on the Battlefield4 dataset which is constructed by images captured from the Battlefield4 game.

Table 1 Quantitative results on the cityscapes validation set

	PQ	SQ	RQ
All	58.0	79.4	71.4
Things	53.4	78.7	67.5
Stuff	61.3	79.9	74.3

Table 2 Comparison with the state-of-the-art methods on the cityscapes validation set

	PQ	PQ_{th}	PQ_{st}	AP	Speed (ms)
Ours	58.0	53.4	61.3	35.0	110
UPSNET [9]	59.3	54.6	62.7	33.0	202
AUNet [8]	56.4	52.7	59.0	33.6	–
FPSNet [6]	55.1	48.3	60.1	–	114
DeeperLab [43]	56.3	52.7	59.0	–	462
PanopticFPN [4]	57.7	51.6	62.2	–	378
SSAP [11]	56.6	49.2	–	31.5	260

9.1.3 Evaluation metrics

To evaluate the performance for panoptic segmentation, we use the Panoptic Quality (PQ) metric [4] which is computed by multiplication of two sub-factors, SQ (Segmentation Quality) and RQ (Recognition Quality), as follows:

$$\begin{aligned} \mathrm{PQ} &= \frac{\sum_{(p,q)\in TP} \mathrm{IoU}(p,q)}{|\mathrm{TP}|} \times \frac{|\mathrm{TP}|}{|\mathrm{TP}| + 0.5 \times |\mathrm{FP}| + 0.5 \times |\mathrm{FN}|} \\ &= \mathrm{SQ} \times \mathrm{RQ} = \frac{\sum_{(p,q)\in TP} \mathrm{IoU}(p,q)}{|\mathrm{TP}| + 0.5 \times |\mathrm{FP}| + 0.5 \times |\mathrm{FN}|}, \end{aligned} \tag{4}$$

where p and g denotes predicted and ground-truth segmentation masks, respectively. |TP|, |FP| and |FN| denote the number of true positive, false positive and false negative, respectively. p is regarded as positive detection when the intersection-over-union (IoU) of p and g is greater than 0.5. We also measure instance segmentation accuracy by averaging over AP^r [42].

9.1.4 Results

We report PQ, SQ and RQ of the propose network for all, things and stuff classes in Table 1. We also compare the performances of our method to the state-of-the-art panoptic segmentation methods in Table 2. For fair comparison, we only report the results using ResNet-50 or ResNet-50-FPN backbones. Although our method does not outperform UPSNet [9], our model is the fastest method among the state-of-the-art methods, while the performance of our method achieves the second-best performance. In comparison with FPSNet [6] which shows similar speed to our method,

the performance of the proposed network gains almost 3 points in terms of a PQ metric. In Tables 3 and 4, we show the quantitative results for Things and Stuff classes, respectively.

We demonstrate qualitative results on the Cityscapes dataset and the Battlefield4 dataset in Figs. 15 and 16, respectively. As shown in Fig. 15, our panoptic segmentation network successfully segments small and thin structures such as traffic light or traffic sign. Also even with low-quality lighting conditions, our method shows good performance on detecting and segmenting objects in simulated battlefields.

9.2 Occluded enemy reconstruction in RTMF

9.2.1 Implementation details and datasets

We train the image completion network using the Battlefield4 dataset which is constructed by the screenshots of the Battlefield4 game. This dataset consists of 1300 images of various poses and situation of soldiers. All images are resized to 256x256 and uses random free-form masks for training. Image completion network is trained on a single GPU with a batch size of 20.

9.2.2 Results

In Fig. 17, reconstruction results using the image completion network are illustrated, where gray regions in the masked images show examples of irregular masks used for training. Even in the case of extreme occlusion, where only a small part of the helmet is visible, the proposed method successes in reconstructing the missing parts well. We also test our method using panoptic segmentation results and show in Fig. 18.

9.3 Agent similarity analysis in HRL

9.3.1 Implementation details and datasets

We adopt visual genome dataset [36] to assign a graph to each agent. To be specific, using the contextual phrases for the single image, at least one phrase is allocated to each agent, some of which has more than two phrase. Then, entities, attributes and edges are extracted from each phrase, which form a graph. Note that even in single image, many distributed graphs can be made because of multiple tangled entities.

Table 3 Quantitative results for things classes on the cityscapes validation set

	Person	Rider	Car	Truck	Bus	Train	m/cycle	Bicycle
PQ	46.2	49.6	65.9	50.0	69.9	59.1	41.5	44.7
SQ	73.6	71.5	84.2	85.9	87.6	82.7	72.4	71.3
RQ	62.7	69.4	78.3	58.3	79.8	71.4	57.3	62.7

Table 4 Quantitative results for things classes on the cityscapes validation set

	Road	Sidewalk	Building	Wall	Fence	Pole	Trafflight	Traffsign	Vegetation	Terrain	Sky
PQ	97.35	72.72	87.09	26.24	27.94	45.29	46.34	63.91	88.98	31.49	86.89
SQ	97.86	83.59	89.41	70.81	71.76	63.57	69.36	76.11	90.02	73.76	92.65
RQ	99.48	87	97.41	37.06	38.94	71.25	66.81	83.98	98.85	42.69	93.79

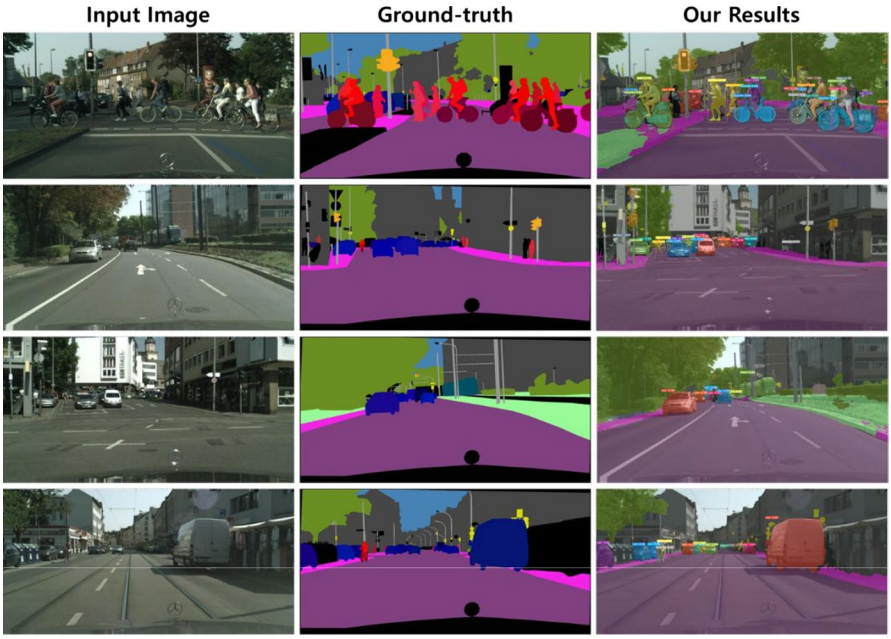

Fig. 15 Qualitative results on the Cityscapes dataset

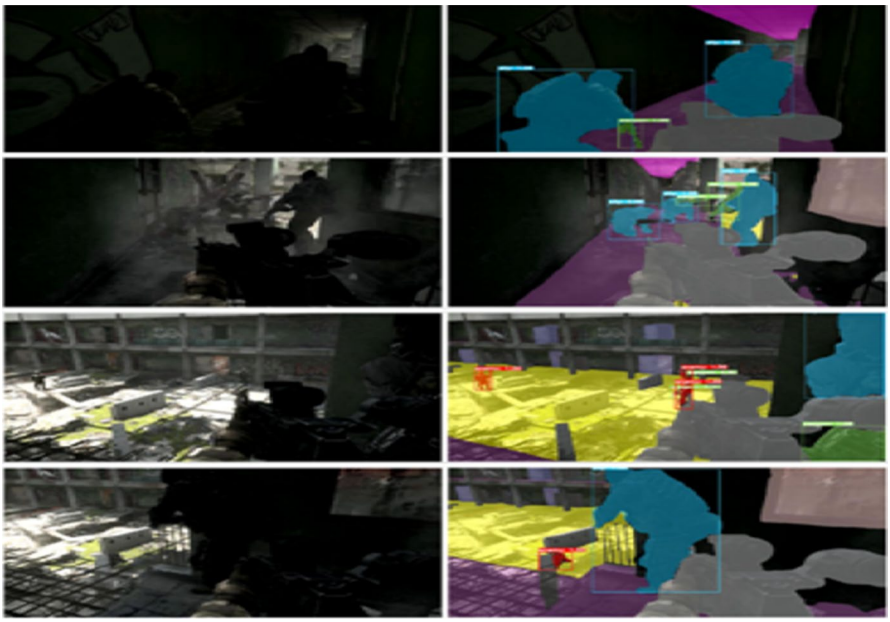

Fig. 16 Qualitative results on the Battlefield4 dataset. Left: input image; Right: panoptic segmentation and detection results

Fig. 17 Reconstruction results on the Battlefield4 dataset

9.3.2 Results

Tables 5 and 6 represent the pairs of agents sorted by cosine similarity of trained agent embedding vector. We considered 26 agents and assigned a graph with more than two nodes to each agent. The number of connected nodes between graphs can be a basis that represents similarity level between agents. So, we can interpret that adjacent graphs shares many nodes. It is observed from Tables 5 and 6 that the trained agent embedding vector has high cosine similarity for adjacent graph and low for dissimilar graphs. By setting a threshold value and classifying the agents, we can create an adjacency matrix of agents (see Fig. 19).

Note that the adjacency matrix in Fig. 19 has relation information between multiple distributed data from each agent, and can be used in our proposed space-time multilayer model for battlefields recognition (STBR) to predict situation labels of the merged graph.

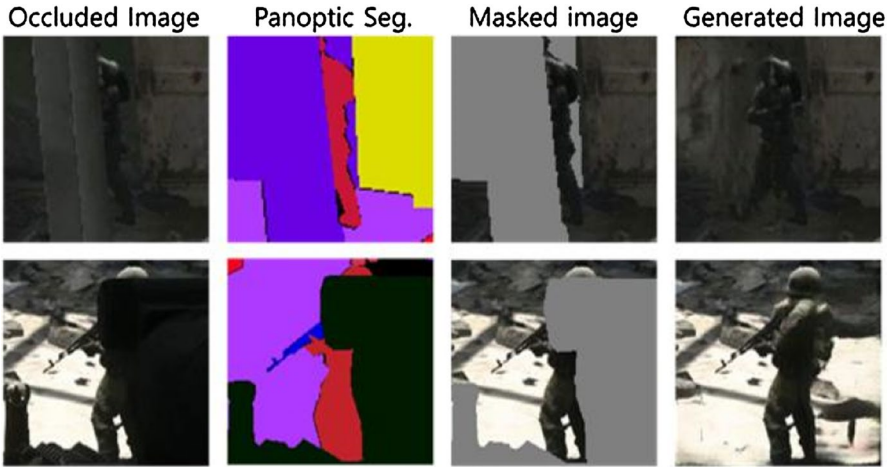

Fig. 18 Qualitative results for reconstructing invisible parts on the Battlefield4 dataset

Table 5 The pairs of Agents whose trained embedding vectors has cosine similarity greater than 0.15

Pairs of agents		Cosine similarity	Number of connected nodes
Agent12	Agent25	0.9547	4
Agent3	Agent26	0.8825	3
Agent17	Agent18	0.7345	3
Agent9	Agent24	0.5296	3
Agent19	Agent22	0.3921	3
Agent2	Agent15	0.2076	0
Agent16	Agent21	0.2048	3
Agent15	Agent22	0.1989	3
Agent14	Agent22	0.1797	3
Agent14	Agent15	0.1571	3

Table 6 The pairs of Agents whose trained embedding vectors has cosine similarity less than -0.3

Pairs of agents		Cosine similarity	Number of connected nodes
Agent15	Agent24	− 0.4137	1
Agent5	Agent16	− 0.3717	0
Agent8	Agent12	− 0.3478	0
Agent8	Agent25	− 0.3450	0
Agent4	Agent23	− 0.3183	0
Agent13	Agent18	− 0.3117	0
Agent13	Agent19	− 0.3080	0
Agent8	Agent18	− 0.3036	0

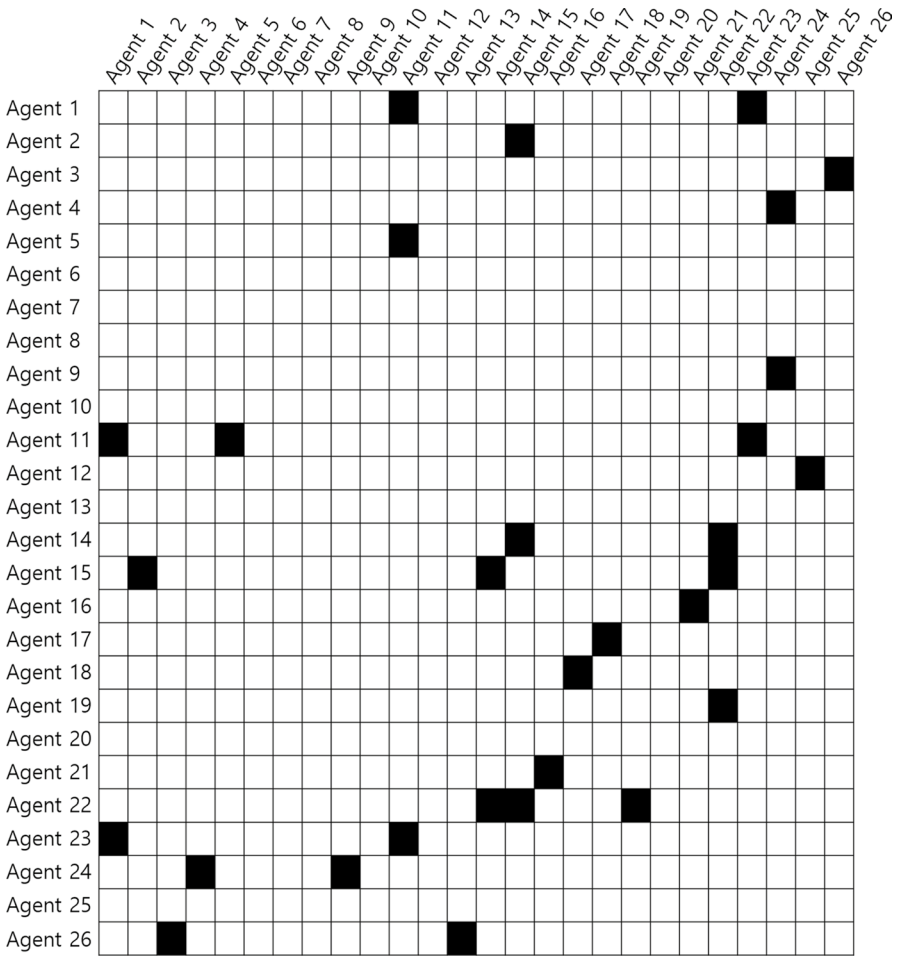

Fig. 19 Adjacency matrix of agents. We set 0.1 as threshold of cosine similarity and adjacent agents are indicated in black in the matrix

9.4 Battle filed awareness in STBR

9.4.1 Implementation details and datasets

Through the battlefield simulation game (Battlefield 4), object learning data was collected to build a YOLOV3-based battlefield situation simulation object learning network. The object was recognized based on the battlefield data from a specific Time Step collected from four allies(agents). We integrated the recognized object information from the screen acquired from our combatants and the location information of our combatants recognized from the map screen to input the deep learning model for situational reasons. The learning data consisted of a total of 250,000 frames of

image data in three battlefield situations (Secure, Vigilant, Engagement), and a total of 1,000,000 frames of image data by using each of the four Agent PCs for one Time Step. We classify data into three battlefield situation; a vigilant situation when enemies, tanks, and helicopters were recognized around the allies, an engagement situation when bombs and explosions were recognized, and a safety situation otherwise.

9.4.2 Results

We measure the average accuracy of situation recognition using servers with the NVIDIA RTX2080. In our simulation, it is observed that average accuracy is 87.248 % for 10,000 frames with 20.45 frame per second (FPS).

10 Conclusion

In this study, we have proposed a Deep AI military staff for supporting commander decision-making by enhancing awareness of combatants. We classify data into three battlefield situation; a vigilant situation when enemies, tanks, and helicopters were recognized around the allies, an engagement situation when bombs and explosions were recognized, and a safety situation otherwise. The proposed model provides collaborative intelligence-based real-time battlefield situation recognition technologies, which is expected to be applicable to actual battlefield environments or combat training simulator. In addition, our proposed models can be used as key technologies not only in the defense area but also in the 4^{th} industrial revolution such as self-driving cars, intelligent robots, smart factory, intelligent security/crime prevention and IoT services.

Acknowledgements This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government (21YR1610, Research on Spatio-temporal Multi-Layer Battlefield Situation Awareness for AI Military Staff) and Korea Research Institute for Defense Technology planning and advancement (KRIT) grant funded by the Korea government DAPA(Defense Acquisition Program Administration) (No. 20-107-C00-008-02, Control technology for collective operation of military ultra-small ground robots).

Data availability All data generated or analysed during this study are included in this published article. For more information on datasets, please see sections. 9.1.2 and 9.3.1.

References

1. Weiss J (2020) The army's ultimate heads-up display: The ivas. https://sofrep.com/news/the-armys-ultimateheads-up-display-the-ivas/
2. DARPA: DARPA demonstrates "Competition" tool at combatant command (2020). https://www.darpa.mil/news-events/2020-03-19a
3. DARPA: Active interpretation of disparate alternatives (AIDA) (2017). https://www.darpa.mil/newsevents/
4. Kirillov A, He K, Girshick R, Rother C, Dollár P (2019) Panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9404–9413

5. Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125
6. de Geus D, Meletis P, Dubbelman G (2020) Fast panoptic segmentation network. IEEE Robot Autom Lett 5(2):1742–1749
7. Kirillov A, Girshick R, He K, Dollár P (2019) Panoptic feature pyramid networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6399–6408
8. Li Y, Chen X, Zhu Z, Xie L, Huang G, Du D, Wang X (2019) Attention-guided unified network for panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7026–7035
9. Xiong Y, Liao R, Zhao H, Hu R, Bai M, Yumer E, Urtasun R (2019) Upsnet: A unified panoptic segmentation network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8818–8826
10. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969
11. Gao N, Shan Y, Wang Y, Zhao X, Yu Y, Yang M, Huang K (2019) Ssap: Single-shot instance segmentation with affinity pyramid. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 642–651
12. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37. Springer
13. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788
14. Barnes C, Shechtman E, Finkelstein A, Goldman DB (2009) Patchmatch: a randomized correspondence algorithm for structural image editing. ACM Trans Gr 28(3):24
15. Bertalmio M, Sapiro G, Caselles V, Ballester C (2000) Image inpainting. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 417–424
16. Satoshi I, Edgar S-S, Hiroshi I (2017) Globally and locally consistent image completion. ACM Trans Gr 36(4):3073659
17. Liu G, Reda FA, Shih KJ, Wang T-C, Tao A, Catanzaro B (2018) Image inpainting for irregular holes using partial convolutions. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 85–100
18. Pathak D, Krahenbuhl P, Donahue J, Darrell T, Efros AA (2016) Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2536–2544
19. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710
20. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864
21. Dong Y, Chawla NV, Swami A (2017) metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 135–144
22. Veličković P, Cucurull G, Casanova A, Romero A, Lió P, Bengio Y (2018) Graph Attention networks
23. Ying R, You J, Morris C, Ren X, Hamilton WL, Leskovec J (2018) Hierarchical graph representation learning with differentiable pooling. http://arxiv.org/abs/1806.08804
24. Zhang M, Cui Z, Neumann M, Chen Y (2018) An end-to-end deep learning architecture for graph classification. In: Thirty-Second AAAI Conference on Artificial Intelligence
25. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778
26. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105
27. Kim C, Li F, Ciptadi A, Rehg JM (2015) Multiple hypothesis tracking revisited. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4696–4704
28. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448

29. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440
30. Bolya D, Zhou C, Xiao F, Lee YJ (2019) Yolact: Real-time instance segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9157–9166
31. Zheng C, Cham T-J, Cai J (2019) Pluralistic image completion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1438–1447
32. Ji R, Chen F, Cao L, Gao Y (2018) Cross-modality microblog sentiment prediction via bi-layer multimodal hypergraph learning. IEEE Trans Multim 21(4):1062–1075
33. Feng Y, You H, Zhang Z, Ji R, Gao Y (2019) Hypergraph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 3558–3565
34. Bai S, Zhang F, Torr PH (2021) Hypergraph convolution and hypergraph attention. Pattern Recognit 110:107637
35. Pohlen T, Hermans A, Mathias M, Leibe B (2017) Full-resolution residual networks for semantic segmentation in street scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4151–4160
36. Krishna R, Zhu Y, Groth O, Johnson J, Hata K, Kravitz J, Chen S, Kalantidis Y, Li L-J, Shamma DA, Bernstein M, Fei-Fei L (2016) Visual genome: Connecting language and vision using crowdsourced dense image annotations. https://arxiv.org/abs/1602.07332
37. Duan K, Bai S, Xie L, Qi H, Huang Q, Tian Q (2019) Centernet: Keypoint triplets for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6569–6578
38. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. http://arxiv.org/abs/1301.3781
39. Wu Y, He K (2018) Group normalization. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19
40. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. http://arxiv.org/abs/1412.6980
41. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3213–3223
42. Hariharan B, Arbeláez P, Girshick R, Malik J (2014) Simultaneous detection and segmentation. In: European Conference on Computer Vision, pp. 297–312. Springer
43. Yang T-J, Collins MD, Zhu Y, Hwang J-J, Liu T, Zhang X, Sze V, Papandreou G, Chen L-C (2019) Deeperlab: Single-shot image parser. http://arxiv.org/abs/1902.05093

Authors and Affiliations

Chang-Eun Lee[1] · Jaeuk Baek[1] · Jeany Son[2] · Young-Guk Ha[3]

Chang-Eun Lee
celee@etri.re.kr

Jaeuk Baek
jubaek@etri.re.kr

Jeany Son
jeany@gist.ac.kr

[1] Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea

[2] Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea

[3] Konkuk University, Seoul, Republic of Korea