OIC: Object Identification using Context for Temporal Tracking in Dynamic Environments

Yasmine S. Antille, Marc E. Solèr
{yasminesheila.antille, marcelias.soler}@student.unisg.ch

23.01.2023

Abstract

Humans have the capability of interpreting our environment based on context and infer knowledge about things we do not effectively see. For machines this becomes a very difficult task to extract (unknown) knowledge from an image, which is a task that research on scene understanding is trying to tackle. Scene graphs extend the notion of individual object recognition by incorporating relationships between objects, allowing broader understanding of scenes, thereby connecting the fields of computer vision and knowledge representation. In this paper we introduce a system for object identification using context (OIC) in dynamic environments. OIC takes images and generates scene graphs to create a scene understanding of the inserted scenario. Then, relying on a symbolic approach, a temporal graph is built which operates on an object and relationship level of all detections over time. We propose Contextual Unique Identifiers (CUIDs), which locally identify objects based on their position, class and direct context and demonstrate high frame-to-frame matching accuracy. The system produces a data-to-text export from the temporal graph for placing queries about the scene, which is accessible via a web interface. We limit the answering capability to just the inserted knowledge and no additional context awareness in order to guarantee that no factually false answers are constructed. The evaluation of OIC shows that it has the potential to be optimized for specific scenarios with some modifications to become a valuable tool for scene understanding in any environment.

Contents

1 Introduction

Humans have a very effective approach to understanding and interpreting their environment combining their direct perceptions with prior-acquired commonsense knowledge. Although the fields of computer science and engineering have challenged and exceeded human capabilities in individual categories such as vision, storing information or rule-based decision making [1], human-like combinations thereof have only been applied successfully recently [2]. Today, scene understanding is at a pivotal point, exploiting combinations of fields as computer vision and knowledge representation.

The mission of extracting and representing knowledge from an image has been studied by the artificial intelligence (AI) community since the beginnings of AI [3]. In logical programming, the use of knowledge bases, specified by logical sentences and combined by inference rules from first-order logic to derive new knowledge, were suggested to store symbolic assertions about the world [1]. With the proliferation of the World Wide Web first ideas to connect knowledge from the web appeared [4], leading to the creation of knowledge bases with connected facts such as Freebase [5] and Berners-Lee's *Linked Data*, suggesting to connect data and information [6]. Google's introduction of their knowledge graph called attention to the topic of knowledge graphs [7] and since then a tide of new research regarding knowledge graphs has been carried out by the the semantic web community. Simultaneously, the publication of *AlexNet* [8] has established deep learning in computer vision, applying data-driven approaches to tasks like image and object classification, object localization, captioning and others [9]. More recently, Johnson et al. [10] extended the task of classifying individual objects to the task of classifying objects and the relationships among each other, represented as a *scene graph*. Scene graphs have been successfully applied for visual-textual transformers, visual question answering, image understanding and reasoning, and detection and recognition of human-object interaction, to name a few main applications [11].

Separately, language models such as *BERT* [12] have shown good performance for predicting the next word in a sentence or question answering. Recently, some of those models have proven to be remarkable at storing and recalling factual knowledge, leading to experimentation of using them as knowledge bases [13] connecting this topic to the one of knowledge bases as introduced above. Conversely, to improve the capabilities of language models in factual question answering, language models have been supplemented by knowledge graphs and converted with data-to-text generation [14], creating further connections to knowledge representation.

In this work, we introduce our system *OIC* (which stands for Object Identification using Context) that tackles exactly this problem. OIC provides a pipeline for identifying and tracking objects dynamically over time. For this, we employ a Scene Graph Generator, which provides classifications of objects and relationships which we then treat as symbolic representations in a graph structure that is conceptually similar to a knowledge graph, a connection already recognized by Zareian et al. [15]. We introduce *Contextual Unique Identifiers* (CUID), which we assign to objects based on their position in the image, their direct context as well as their class, and show that these achieve a high frame-to-frame matching accuracy. We use a temporal graph as knowledge base to keep the identifying properties of each object as well as a history of the relationships between the objects and provide a data-to-text export to integrate our temporal graph into language models. As a final step, we show that a language model supplemented with the synthetic data-to-text export serves as a natural language interface to answer simple queries about facts stored in the temporal graph. Ultimately, we consider our system as a general purpose framework for dynamic object identification and tracking that can be adapted for special purposes. To customize OIC, we discuss several modifications necessary and define touch-points in our implementation.

2 Related Work

Currently, computer vision developments are strongly supported by deep learning approaches as reliable simple object detection and recognition in visual understanding tasks is no longer an unsolved problem [11]. A higher-level visual understanding and reasoning is necessary to capture relationships between objects in a scene as a motivation. In order to achieve this, a data structure describing object entities in a scene and the relations to other objects is necessary. Research in

this field is relevant for many advanced applications, especially in dynamic and challenging environments. In the following we will refer to important resources describing this shift from object recognition for tracking purposes to scene graphs as a semantically enriched approach towards scene understanding. Furthermore, the challenges of natural language descriptions from such a graph are related to other researchers' work.

Object tracking and identification Tracking and identifying objects, in general, is a challenging problem and increasingly difficult in a dynamic environment. To name a few factors: abrupt object motion, changing appearance patterns, object-to-object and object-to-scene occlusions as well as camera motion pose difficulties to tracking objects, which usually demands the location and/or shape of an object in every frame [16]. Numerous approaches for object tracking have been brought forward, many of which use information from a single frame. A notable approach to object detection is *Background Subtraction*, which became popular following Wren et al.'s publication [17]. Here, a representation of the scene, called the background model, is built and then deviations from the model for each incoming frame are found in order to determine moving objects. Most of these approaches are pixel-based, meaning that they operate solely on numerical inputs and do not take contextual and semantic information into consideration.

Scene graphs While the ability to track and identify objects in a dynamic scene is improving rapidly, a higher level of recognition is needed for complex scene understanding. Exploring the rich semantic relationships between objects in a scene is a more challenging task, fostered by the lack of proper data representation [18]. Johnson et al. [10] proposed a visual representation for this task, called the *scene graph*, which is a visually-grounded graph over objects detected in a scene. The nodes correspond to the object bounding boxes, and the edges represent their pairwise relationships. Gu et al. [19] declare that there are two categories in which current approaches for scene graph generation fall. The first category comprises methods which detect objects first and then recognize the relationships between them, such as [20] and [21], and the second contains methods which jointly identify the objects and their relationships based on object and relationship proposals [22, 23]. Scene graphs have since been improved [24] and used in combination with commonsense knowledge graphs to improve accuracy [15].

Scene graph generation The authors Cong et al. [25] presented a new method to generate scene graphs called Relation Transformer (RelTR). RelTR is an end-to-end model that infers a fixed-size set of subject-predicate-object triplets from an image or video. This method is designed to predict sparse scene graphs directly from the visual appearance of the objects, without the need of combining all entities or labeling all possible predicates. The authors claim that their method outperforms existing scene graph generation methods with high-quality results and faster inference times, making it suitable for our application.

Question answering Ever since the use of chatbots has found footing in numerous fields [26], society's interest in asking systems specific questions regarding the system's target tasks and instantly receiving answers has increased greatly. GPT-3, bloom, and Roberta are large language models that have recently demonstrated their usefulness in explainable AI, information retrieval and more, even though they have many limitations and not always provide correct information [27]. GPT-3 (Generative Pre-trained Transformer), for example, is a language model that uses deep learning to produce human-like text, and it is intuitive that large amounts of data are needed for training to produce relevant results. Here specifically, the language model is trained on an unlabelled dataset including texts from Wikipedia and many other sites [28], which gives it its strong suitability for text generation on related question input. In contrast to models such as BERT [12] (as mentioned in the introduction) or GPT-3, ALBERT [29] has fewer parameters (millions versus hundreds of millions in BERT and hundreds of billions in GPT-3), has notably smaller memory consumption, faster evaluation times and can therefore be operated on desktop computers.

Data to text Although many tasks in machine learning target the extraction of structured data from unstructured data, such as extracting tabular data from documents, attention has been called

to the inverse task of generating text from structured data, or data-to-text, as for example in Wiseman et al. [30]. The task is traditionally accomplished with fuzzy matching [31], and more recently by using neural networks with a hierarchical model [32]. While typical applications of data-to-text were the representation of data in a more human-readable fashion, its usefulness was recognized for feeding data in knowledge bases to language models to improve them in answering questions with more explicit rather than latent knowledge. A prominent example for this is TEKGEN, utilizing fine-tuned neural networks supplemented by heuristics for the purpose of extending a comprehensive language model [14].

3 Overview

OIC can be categorized into 3 main components and contributions: (1) the Scene Graph Generator, which takes images from a video feed and creates scene graphs for each frame, (2) the Symbolic Link, which is the linking step between the generated scene graphs and the Temporal Graph, and (3) the Querying Interface, where a user can interact with the Temporal Graph knowledge. A high-level overview of the system's process is listed in Algorithm 1. The Scene Graph Generator is a modified version of RelTR [25], written in Python and PyTorch. It is publicly available at `https://github.com/macemoth/RelTR`. The Symbolic Link and the Scene Querying interface are both implemented in Python, while the Scene Querying interface utilizes the ALBERT language model [29]. The source code for this project is available at `https://github.com/Interactions-HSG/IMP-OIC`. The individual components will be discussed in the following chapters.

Algorithm 1 High-level algorithm of the system's procedure

function MAIN(I) ▷ I is a set of images
  $t \leftarrow 0$ ▷ t is the current timestep
  while $I \neq \emptyset$ **do**
    $\text{Frame}_t \leftarrow I(t)$
    $\text{Triples}_t \leftarrow$ SCENEGRAPHGENERATOR(Frame_t)
    $\text{FrameGraph}_t \leftarrow$ CREATEGRAPH(Triples_t)
    $\text{TemporalGraph}_{t+1} \leftarrow$ INSERTFRAMEGRAPH(TemporalGraph_t, FrameGraph_t)
    $t \leftarrow t + 1$
  end while
  NLDescription$\leftarrow$ DATATOTEXT(TemporalGraph_t)
  return TemporalGraph_t, NLDescription, Visualization
end function

4 Scene Graph Generator

In the course of the project, rather than just relying on mere object detection algorithms such as YOLO, we made the decision to capture a scene with objects and their relationships. Given an image, a symbolic, graphical representation can be extracted, where every node represents an identified object and every edge represents the relationship between the objects. This leads to the possibility of understanding and interpreting a scene and leveraging the semantic relationships for object identification and tracking.

4.1 Individual object detection versus object-relationship detection

At the beginning of the project we evaluated the individual object recognition approach against scene graphs. Individual object recognition systems are attractive as they feature both high recognition accuracy and frame rates and the robust implementations. While scene graphs provide the position, class label and relationships for further processing, relationships are missing from the individual object recognition. In our implementation with the YOLO system, we used spatial clustering as proxy for relationship edges. However, this approach was limited in precision and recall.

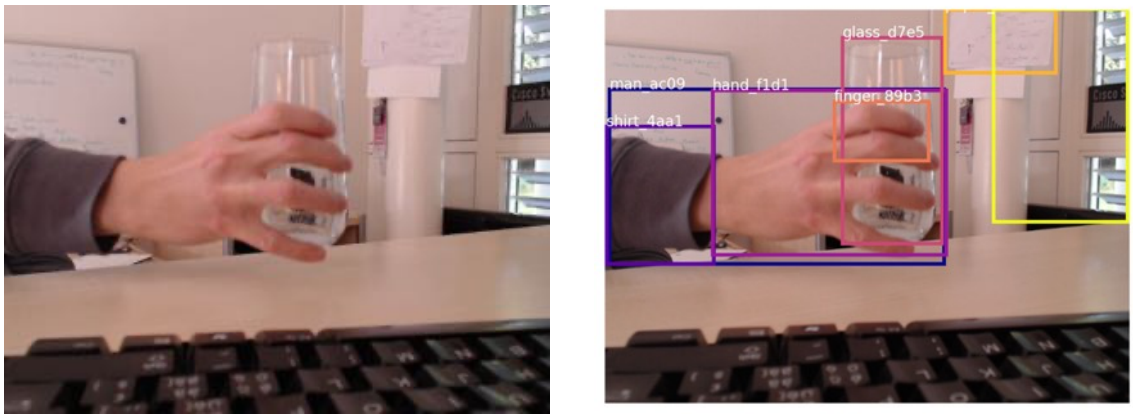

Figure 1: Visualised sparse scene graph of a simple image scenario. Boxes are coloured differently for each identified object. Triplet proposals are not visualised here.

In terms of recall, objects in proximity were considered to be in a relationship to another although they may not actually be. E.g. objects that are actually in the fore- and background of a scene and share no relationship suddenly appear to do so, since the depth is lost in the 2-D projection of an image. This becomes especially apparent for moving objects as their spatial neighbors change frequently (and are thus falsely interpreted as belonging to the context) while the semantic context roughly remains identical. Conversely and in terms of recall, relationships of objects that were spatially separated were not recognized. Meanwhile, our system provided considerably superior results by simply including the relationships from the scene graph. Although scene graph generators have notably lower accuracy as individual object recognition [15], its results appeared more promising.

A strong disadvantage of scene graph generators is the higher computational load, reducing the frame rate. Additionally, scene graphs require more additional relationship training data than individual object recognizers, and there exist only few datasets. As scene graph generation is an active recent research area, implementations have proven to be less robust and more focused on performance evaluation. As the system provides an interface to place queries about the scene, the additional relationship labels should improve search performance and provide richer results that are not available from individual object labels. Furthermore, as OIC tolerates images that have been taken with up to several seconds difference and feature notable changes in the scene, frame rate requirements can be relaxed, compensating the higher evaluation time of the scene graph generator, in particular for modest hardware[1].

4.2 Generating the scene graphs

Once the decision to go forward with a scene graph generation was made, a state-of-the-art approach was chosen as basis. Integrating the RelTR method (see Chapter 2), we obtain a scene graph containing k relationships found in the image. The amount of predicted relationships is a fixed-sized set based on the given fixed amount of subject and object queries using attention mechanisms. For our prototype, we use $k \in \{5, 6, 7\}$, which we have found by qualitative evaluation. The scene graph is exported as a set of triples in the form (s, p, o)[2] e.g. `(man, holding, glass)`. In Figure 1 a visualisation of the detected and identified objects can be seen. An example triple defining a detected relationship visualised in the image can be seen in Listing 1, which describes the predicted (s, p, o) structure. Some predictions can be duplicated in meaning (e.g. `(man, has, hand)` and `(hand, of, man)` and sometimes RelTR cannot label very difficult relationships correctly (e.g. `(glass, is, hovering)`. Nevertheless the generated results are of high quality.

[1] On appropriate hardware, RelTR evaluates at 16 frames per second [25], while one frame per second is sufficient for most scenes.

[2] s = subject, p = predicate, o = object

Listing 1 JSON file extract of detected triples from scene graph represented in Figure 1 proposing the *(man, holding, glass)* objects relationship.


```
{
    {"subject": {
        "id": "hand",
        "xmin": 130.552734375,
        "ymin": 97.84503173828125,
        "xmax": 415.457275390625,
        "ymax": 298.21221923828125
    },
    "predicate": {
        "id": "holding"
    },
    "object": {
        "id": "glass",
        "xmin": 289.2110595703125,
        "ymin": 33.39134216308594,
        "xmax": 410.52490234375,
        "ymax": 284.7768249511719
    }
}
```


4.3 Limitations

RelTR is a research implementation and therefore not performance optimized. By adding such optimization (e.g. using a production-oriented neural network library, re-using scene graphs from previous frames, or by exploiting a hierarchical entity model as in YOLO), RelTR's computational requirements may be reduced. Also, the general-purpose pre-trained model we use in the prototype could be replaced by a special-purpose model that is concentrated more on relevant content of a scene.

5 Symbolic Link

The Symbolic Link represents an extension of the Scene Graph Generator operating on an object and relationship level instead of the pixel level. It fulfills the following tasks: (1) construction of a graph (FrameGraph$_t$) from triples obtained from the Scene Graph Generator, (2) construction of a temporal graph (TemporalGraph$_t$), which includes the creation of CUIDs and re-identification of CUID-identified objects in new frames, and (3) the data-to-text export of the temporal graph that is later used for the natural language interface.

5.1 Creating frame graphs

Constructing the frame graph from triples in the form (s, p, o) is ambiguous because a recognized entity in an image may be part of several triples, as mentioned as a limitation above. For instance, in `(woman, holding, phone)` and `(phone, on, table)` the `phone` may be the same entity, or there may be several phones in the scene. Consequently, the location of the entities has to be examined next to the class name. For this, we define a similarity metric dependent on the bounding boxes and consider two bounding boxes identical if their similarity metric is within $\varepsilon < 0.3$ from a congruent score of 1. The similarity metric is defined as overlap of the bounding boxes normalized by the area of the larger bounding box. Compared to simple overlap, our metric accounts for size differences that arise, e.g. for small entities in the foreground of a large entity.

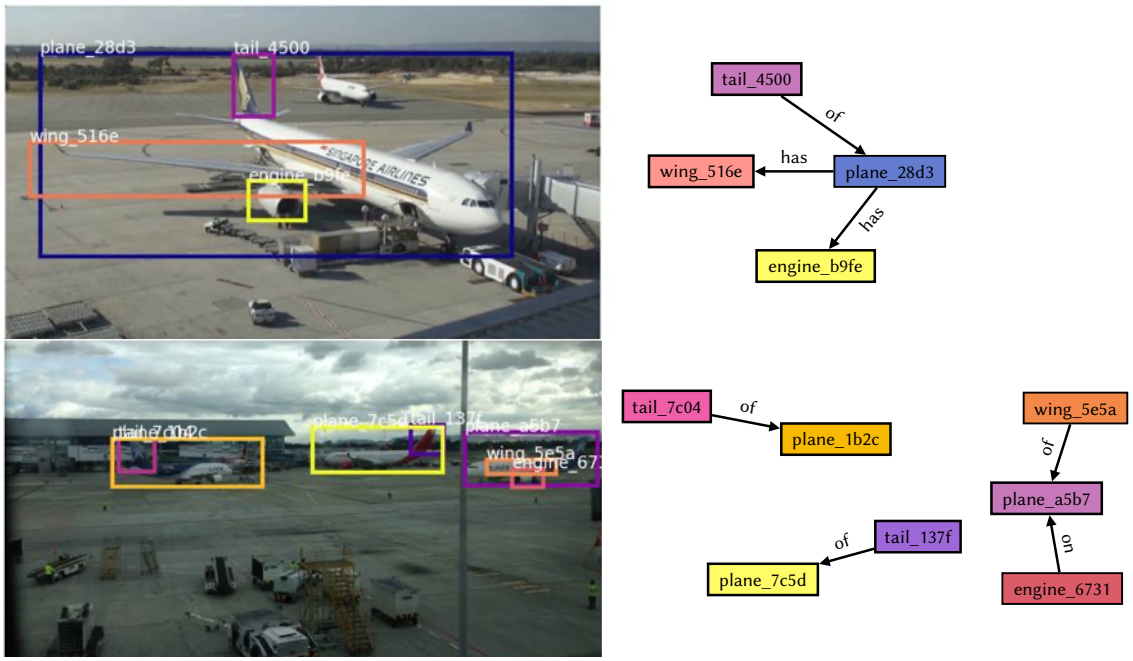

Figure 2: Top: Individual frame with the overlay of the identified objects and the corresponding slice from the temporal graph. **Bottom**: The planes in the scene each carry another CUID, although their class label is the same. Objects in their context such as wings and engines are used to identify and track the planes (and vice-versa).

In OIC the concept of contextual identity represented by the CUID is crucial. Identification shall therefore be distinguished from object classification, studied and provided by systems such as *You Only Look Once* (YOLO) [33] that receive an image as an input and provide a list of class labels and their positions within the image. In this work identification aims to assign labels in a scene to entities uniquely, such that for example two entities can be distinguished from each other both when they are in a scene at the same time or at different times (see Figure 2). In contrast to appending a number to classes on top of a classification framework (e.g. `car1`, `car2`) to distinguish them, we obtain more depth by anchoring the identities within the scene graph. Concretely, we derive identity from (1) the entities' neighbors in the scene graph, (2) the location inside the image and (3) their classification label.

5.2 Identification and constructing the temporal graph

OIC is time-sensitive and in addition to distinguishing entities and issuing CUIDs, it is able to re-identify previously identified entities in similar images, such as subsequent frames from a video feed. Most of this behaviour is implemented in Algorithm 2, or step (2) (as introduced in the beginning of this chapter). In the Symbolic Link, this the most complex component and deserves a more detailed explanation. The algorithm's inputs are:

- the current temporal graph TemporalGraph_t at timestep t,
- a frame graph FrameGraph_t of the current frame,
- a weighting parameter $\alpha \in [0, 1]$, and
- a threshold for minimum confidence Confidence_{min}.

From line 3 on, we iterate over all nodes from the frame graph f, for which we either seek a similar node in the temporal graph $g \in G$ which we can update (line 10), or create a new node in the temporal graph (line 7). The decision is dependent on the similarity score (which we term confidence) between f and g, which is determined by the number of common neighbors

Algorithm 2 Algorithm for inserting a frame graph into the temporal graph

function INSERTFRAMEGRAPH(TemporalGraph$_t$, FrameGraph$_t$, α, Confidence$_{min}$)
  $G \leftarrow$ TemporalGraph$_t$.nodes ▷ We store the nodes in a set to reduce the search space after successful matches
  for $f \in$ FrameGraph$_t$.nodes **do**
    for $g \in G$ **do**
      Similarity$_f(g) \leftarrow \alpha$ NEIGHSIMILARITY$(f, g) + (1 - \alpha)$ SPATSIMILARITY(f, g) + CLASSSIMILARITY(f, g)
    end for
    if $\max$ Similarity$_f(g) <$ Confidence$_{min}$ **then**
      CUID$_f \leftarrow$ GETCUID(f)
      TemporalGraph$_t$.add(f, CUID$_f$)
    else
      BestMatch$_f \leftarrow \operatorname{argmax}_g$ Similarity$_f(g)$ ▷ The node in the temporal graph that is most likely to be the predecessor of f
      TemporalGraph$_t$.update(BestMatch$_f$, f)
      $G \leftarrow G-$ BestMatch$_f$ ▷ Remove the best match from G as it is assigned
    end if
  end for
  for $e \in$ Framegraph$_t$.edges **do**
    if $e \in$ TemporalGraph$_t$.edges **then**
      TemporalGraph$_t$.append(e)
    else
      TemporalGraph$_t$.add(e)
    end if
  end for
  return TemporalGraph$_{t+1}$
end function

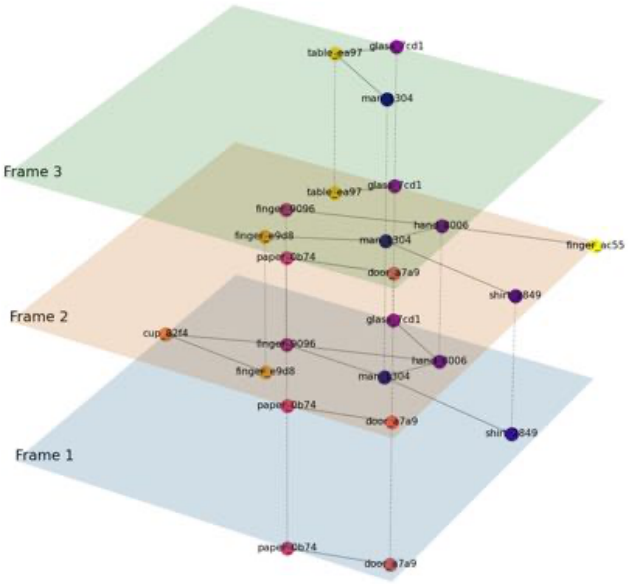

Figure 3: Visualization of a temporal graph. Each frame is represented by a plane, on which the temporal graph at time step t is laid out. Dotted connections represent a re-identification.

(NEIGHSIMILARITY), the spatial similarity of their bounding boxes (SPATSIMILARITY) and the class similarity (CLASSSIMILARITY). All three similarity scores yield values in the range $[0, 1]$. A parameter α closer to 1 increases the importance of neighbor similarity where more consideration is given to the spatial similarity with α closer to 0. In our implementation, we use a strong emphasis on local similarity with $\alpha = 0.3$, as we expect objects to move smoothly from frame to frame. When more movement of objects is expected between frames (e.g. when the frame rate is low), a stronger emphasis on neighbor similarity is advised. In the first graph, where the temporal graph is empty, all objects from the frame graph receive a default confidence of -1 and are inserted into the temporal graph.

For Confidence$_{min}$, we set a value of ≈ 0.6, requiring a confidence of at least this value to re-identify a newly recognized object as a known one in the temporal graph. The update operation of a node in the temporal graph (line 10) updates the bounding box of the object to allow spatial tracking and sets a flag that the object was present in time step t. Although this decision keeps the graph larger than an alternative approach of storing frame nodes and the connections of objects with the same CUID, we can easily re-identify objects that have not been seen for any number of frames. In line 2, all nodes of the temporal graph are stored in a separate set G, reducing the set of nodes G to be iterated in the inner loop. This greedy algorithm ignores the comparison of some f with gs that have already been matched. This provides a worst-case complexity of $\mathcal{O} = |F| \cdot |G|$ and an average complexity of $\mathcal{O} = |F| \cdot 1/2|G|$ where $F =$ FrameGraph$_t$.nodes. This procedure corresponds roughly to the first step of the Hungarian assignment method [34]. An obvious improvement would be to implement this method as whole, but for clarity and in relation to the much higher computational requirements of RelTR, this has been left out. As a final step, the edges of the frame graph are inserted into the temporal graph. Through the previous steps, all nodes from the frame graph exist in the temporal graph, so the edges can either be created (line 20) or appended (line 18). In difference to the node insertion, existing edges are not updated, but appended to an edge list to store the relationship history of the scene. The nodes of the temporal graph represent all objects that have been detected over the complete time span. With this step, TemporalGraph$_t$ is completely updated by FrameGraph$_t$ and returned as TemporalGraph$_{t+1}$. A complete view of an example temporal graph is rendered in Figure 3.

Our approach has in principle several advantages compared to existing frameworks. In terms of object tracking, this symbolic approach relies on context (via neighbor and class similarity) next to location in the image, making it more robust than sole pixel- or location-based approaches. It could

further be argued that a conventional object tracking framework may be trained on more specific instances, working on a finer class granularity and able to distinguish similar, but distinct objects in a scene. Although this approach may work for some situations, our approach has the advantages that (1) even objects that have the identical pixels are recognized as distinct as their neighbors and positions differ and (2) our system is able to provide a human-understandable explanation on the objects' identities, consisting of the objects' context and relative positions.

5.3 Choosing the abstraction layer

At this point, it is appropriate to mention the relation of our approach to the area of *Neurosymbolic Programming* that aims to combine subsymbolic deep-learning approaches with symbolic operations [35]. From this view, the Symbolic Link operates on a higher abstraction level as the the Scene Graph Generator that utilizes neural networks. In difference to a typical Neurosymbolic system that draws from established tools such as logical programming and domain-specific languages, we provide a less general and more customized approach to symbolic manipulation.

In our case, the separation between the subsymbolic and symbolic manipulation is trivially given by the interface to the scene graph generator, but may be altered in both directions. On one end, it would be possible to hand over more reasoning to the symbolic part by increasing the resolution of the scene graph generator, e.g. by recognizing a hand as grouping of fingers, leaving the interpretation of them as a hand to the symbolic system and giving it more anchors for identification (e.g. recognizing the individual letters of a sign instead of the sign as a whole). Although this may increase explainability of the system, more objects had to be stored in the temporal graph and high sensibility of the neural part may be confused by image artifacts. On the other end, the scene graph generator could be trained on more specific objects, such as to recognize an airplane of a particular company due to the number of engines, and the logo etc., surrendering more reasoning to the neural part. Problems may include an overfitting of the neural part and less explainability, as the recognition mechanics in the scene graph generator are latent, but symbolic search would become easier. Our approach is located in between both ends, but still attempts to exploit one advantage of the first approach, namely to use an object's neighbors as "anchors" for identification, as explained in this chapter.

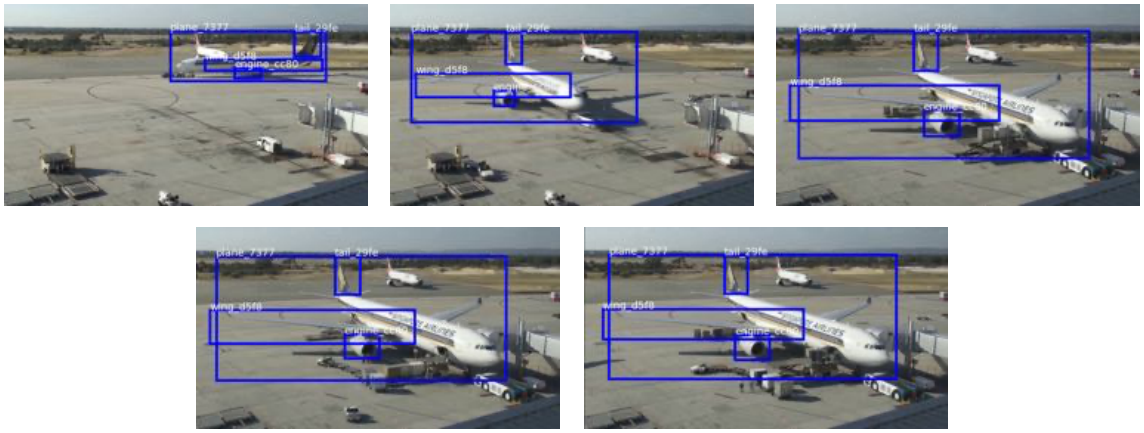

Figure 4: Qualitative results of the scene graph generation using an airport scenario of an airplane docking and being unloaded. The blue boxes represent the identified objects. In this setting, a limitation is the inability of RelTR to recognize the second airplane in the background due to the attention on the airplane in the foreground.

5.4 Evaluation

To evaluate our system, we chose a qualitative approach of the scene graph generation and picked two substantially different scenarios to highlight different central achievements of our system. Our

evaluations were typically done by visual inspection of the identification logs and the images, as discussed below.

In Figure 4 we introduce a sample scenario at an airport, where an airplane rolls up to its gate and the luggage is starting to be offloaded. We see the resulting annotations of the Scene Graph Generator. For the evaluation we chose a very low frame rate (<1 fps), and it is clearly visible that our system proves to be robust in recognizing and identifying the same objects even in subsequent frames that show big leaps in time (with moving objects). Therefore, we can conclude that it is not required to provide a high frame rate to OIC and dropped frames are often tolerated well by the system.

For a rather contrasting scenario from this airport example we evaluate a different setting, which we will discuss in more detail and to also showcase some generalisability. In Figure 5 another evaluation scenario is shown, where we highlight the re-identification of an object that is removed from the scene and re-introduced at a later point of time. This worked for multiple entities, namely the `woman`, `glass` and `bottle`, which can be inspected by having a close look of the top right and bottom left annotated images. The 6 images shown are not all frames from the system, but a selected view to display here. We prove that our system is capable of matching the entities with a high enough confidence score to associate them with the same identity even if objects leave the scene over multiple frames and later reappear. Another notable outcome in this example is that identical bottles are distinguished from each other and based on their individual (and markedly different) context can be kept separate even if removed. See Figure 7 in Appendix A for the visualisation in steps of 10 frames to keep it assessable. The re-identification of `woman_b8f5` for example is apparent as it is not detected in Frame 20. Listing 2 shows an excerpt of the console output of the symbolic link step as reference.

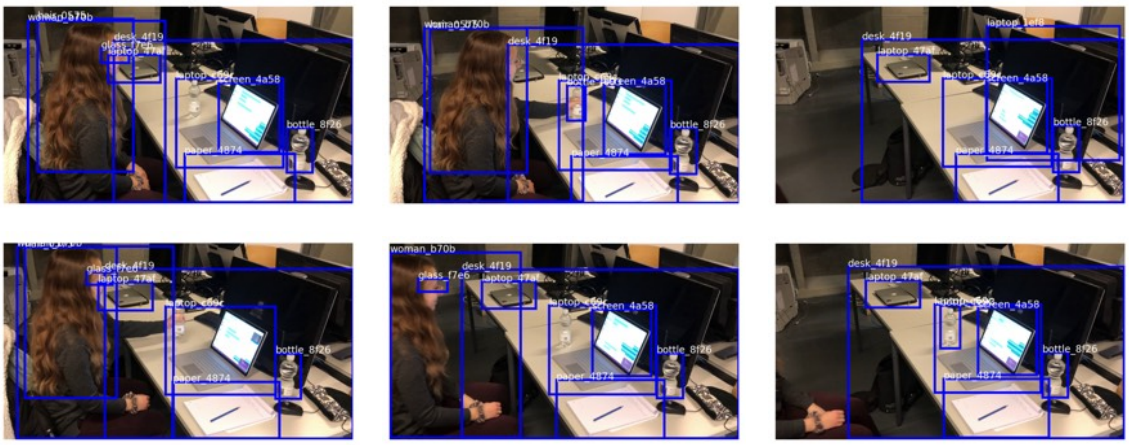

Figure 5: Qualitative results of scene graph generation on desk with twin bottles scenario. A woman removes a bottle from the captured environment and reintroduces it and it get recognized and identified as the same object.

6 Scene Querying

The temporal graph can become complex with a large number of frames and a visual inspection becomes unsuitable for all but simple examples. Therefore, an efficient interface for placing queries about the graph is required. The temporal graph can be recognized as a knowledge base, containing information about objects and their relationships in space, context and time. Traditionally, such knowledge bases may be queried using relational query languages such as SQL. For graph structured knowledge bases, query languages like SPARQL [36] or Cypher [37] are frequently used. However, these languages typically expect the user to have some knowledge about the data scheme (for relational databases), or the node and edge types (for graph databases), in order to write their queries. In contrast, natural language search interfaces such as the one used by Google [38] require the user to only have superficial knowledge of the queried data structure. The user

may use the information obtained from a query to improve the next query, "hopping" to the desired result.

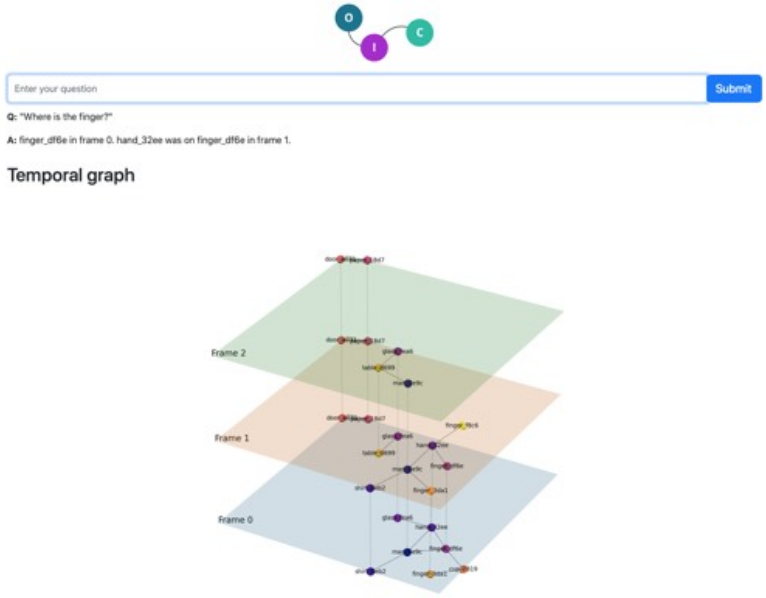

Figure 6: The query interface. The search bar is combined with a 3-d visualization of the temporal graph to give the user a high-level overview of the scene.

We deem the temporal graphs generated by our system as good candidates for a natural language interface because the user may only have superficial information about the scene (e.g. knowing it describes an airport) and may not know an exact relationship nor any CUIDs to query for. Thus, by placing multiple queries, the user can learn more about the scene, e.g. CUIDs or frames, that they can then reuse for subsequent, more specific queries.

To this end, we employ ALBERT, a pre-trained lightweight language model that can be used as question answerer. As question answerer, ALBERT works as follows: first, a general-purpose pre-trained model is loaded. Then, two natural language strings are inserted, the first one being the context and the second being the question. The language model then yields start and end indexes, marking the answer within the context string. In comparison with models such as GPT-3 or ChatGPT, the answer may only be a selection of the context, and not constructed separately. Although this restricts the model's inference capability, it also guarantees that no false answers are constructed, a typical limitation of large language models [27]. In worst case, our system gives irrelevant rather than incorrect information.

Typically, the context given to the model is a text of several sentences. For our application, we use a natural language data-to-text export of the temporal graph as context for the language model. The data-to-text conversion is done as follows: For each edge in the temporal graph (recall that we store an edge between nodes n_1, n_2 for time step t when the nodes have a relationship in the frame at t) we create a sentence in the form (n_1, p, n_2) in frame t, e.g. `engine_1785 was on wing_6f9d in frame 2`. We use a hand-crafted lookup table to convert simple predicates as `"belonging to"` to `"belonged to"` to give the language model temporal cues. This procedure is a significantly simplified version of the TEKGEN method that is designed for knowledge graphs, incorporating multiple edges at once and reformulating the sentences to sound more natural [14]. However, the model still has severe difficulties with recognizing the temporal relationships in the data-to-text export and is confused by relationships present in several frames. Therefore, we restrict every relationship to only be featured once in the export.

Our query interface is available via a web interface (see Figure 6), where additionally a 3-D visualization of the temporal graph is displayed, intended to give the user a high-level overview of the scene and several CUIDs to start with. Currently, as mentioned above, the querying capabilities are limited to the small amount of information introduced from the temporal graph and the restricted language model. Expected questions that the system should be able to answer are re-

garding facts that are inferred from the imported triplets. Question-answer scenarios may include:

- Open question: **Q**: *What is on the desk?* **A**:*laptop_0af6 was on desk_a92e in frame 0. paper_93fe*
- Closed question: **Q**: *Where is the bottle?* **A**: *desk_a92e*
- Relational question: **Q**: *Who has hair?* **A**: *woman_29ee*

Questions about temporal facts (e.g. "For how many frames was the laptop on the desk?") cannot be answered with the current model as this would require language model to formulate a new sentence and have a notion of time, what it is currently both lacking[3].

7 Optimizations

OIC is designed to be a general purpose system and neglects particular use cases in several aspects. In this paragraph, we describe immediate modifications that can be implemented for adapted and improved performance of OIC in these use cases. RelTR, our scene graph generator, is pre-trained on the Visual Genome dataset [39] containing everyday scenes and objects. In order to recognize objects and relationships from other use cases, we suggest to train it on specific data from these. Instead of re-training RelTR completely and losing the general purpose capabilities, the current model can be used as a checkpoint on which to continue training. For us, training RelTR proved to be a challenge, as relational data is seldom found in image datasets and may need to be created by hand. This is due to the fact that RelTR requires several adaptations in its codebase, as it contains several references to the Visual Genome dataset. As research progresses, we expect production-level implementations in the next years. On the other end, the ALBERT model of the query interface also utilizes a pre-trained language model and is fine-tuned on the SQuAD1.1 question-answer model [40]. Two modifications are possible:

1. to improve overall performance of the model, a larger BERT model may be utilized. However, this increases the computational and memory load, leading to longer execution times.
2. The other modification is to fine-tune the current ALBERT model to perform better on questions from the use case [4].

As mentioned above, none of the examined language models were able to properly recognize temporal relationships. To the best of our knowledge, language models have not been optimized for the use with temporal graphs yet, although the combination with knowledge graphs has been made. We deem research in this area as an interesting future direction considering that much of human knowledge is historical and thus, temporally related. More generally, OIC can be adapted for different kinds of scenes, e.g. with faster movement or very similar objects, by adjusting the α and Confidence$_{min}$ parameters described in Section 5.

8 Conclusion

In this paper we introduced OIC, a novel system for temporal object identification and tracking using context in dynamic environments. With a state-of-the-art scene graph generator and natural language processing we combine methods and ideas from different fields in computer science to form a general purpose scheme, which can be optimized for specific scenarios. Object detection algorithms alone could already provide a baseline for the querying of the scene, however, we evaluate scene graphs to be the better choice as more detailed and structured representation of the objects lead to better identification results, especially in dynamic settings.

The symbolic link is able to issue CUIDs to objects based on their location, semantic context and class label, and can re-identify objects even in dynamic scenes with low frame rates. Although

[3] We evaluated GPT-3 with the same context and did not receive satisfying results for temporal questions, suggesting that language models have trouble reasoning with time.

[4] Instructions for fine-tuning are available at `https://www.tensorflow.org/tfmodels/nlp/fine_tune_bert`.

the recall and precision of the used scene graph generator is moderate, the symbolic link can (re-)identify and track objects in a qualitatively satisfying manner relying only on symbolic input and provides textual and visual outputs.

During evaluation of the scene graph generation and symbolic linking steps we observed that in scenarios with many similar entities, our system becomes more easily confused and depending on the Confidence$_{min}$ parameter either identifies new objects as already known, or known ones as new objects. Often, these problems could be solved by varying Confidence$_{min}$. In other situations, the results from the scene graph generator exhibited strong variance with objects and relationships alternating their presence in subsequent frames.

Finally, a simple language-model-based query interface allows to make simple queries about a scene formulated in natural language. Although the used model has moderate performance, recent evidence shows that scaling the model may improve performance significantly [41] and provides a basis for further work in connecting (temporal) knowledge graphs with language models. Therefore, an interesting enhancement of the system would be to introduce more background (or rather commonsense) knowledge about the scene environment that will be introduced to the natural language processing, so that the answers can elaborate more on the context. This way even a bit more far fetched questions that are not factually stated in the scene graphs can be asked, such as *"What kind of work has been done on airplane_x?"*[5] (e.g. deicing, container unloading, refuelling, etc.) or *"Where did airplane_x dock at?"* for the airport scenario and *"What did the woman move from the desk?"* for the bottle scenario. We believe that this may be achieved by leveraging language models integrating (commonsense) knowledge graphs, similar to Agarwal et al. [14].

Further research on top of this application may include deeper comparison studies regarding existing scene explanation systems in literature and applied scenario studies that explore the system's efficiency, user experience and acceptance of the question answering capabilities.

Acknowledgments

We would like to express our sincere gratitude to Professor Simon Mayer for his invaluable support, patience and guidance throughout this project. We also sincerely thank Sanjiv Jha and Kimberly Garcia for their support, strong encouragement and helpful critique to our discussions. Finally, we want to acknowledge our colleagues for valuable exchanges and useful input and advice for our project.

[5] *airplane_x* referring to an identified object that must be given by the projected scene graph

References

[1] S. J. Russell and P. Norvig, *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] J. McCarthy *et al.*, *Programs with common sense*. RLE and MIT computation center Cambridge, MA, USA, 1960.

[4] D. W. Hillis, ""aristotle" (the knowledge web)," 2000. `https://web.archive.org/web/20070224063818/http://www.edge.org/3rd_culture/hillis04/hillis04_index.html` [Accessed on 12.01.2023].

[5] J. Menzel, "Deeper understanding with metaweb," 2010. `https://googleblog.blogspot.com/2010/07/deeper-understanding-with-metaweb.htmll` [Accessed on 12.01.2023].

[6] T. Berners-Lee, "Linked data," 2006. `https://www.w3.org/DesignIssues/LinkedData.html` [Accessed on 12.01.2023].

[7] A. Singhal, "Introducing the knowledge graph: things, not strings," 2012. `https://blog.google/products/search/introducing-knowledge-graph-things-not/` [Accessed on 12.01.2023].

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[9] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[10] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3668–3678, 2015.

[11] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "Scene graphs: A survey of generations and applications," *arXiv preprint arXiv:2104.01111*, 2021.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[13] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?," *arXiv preprint arXiv:1909.01066*, 2019.

[14] O. Agarwal, H. Ge, S. Shakeri, and R. Al-Rfou, "Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training," *arXiv preprint arXiv:2010.12688*, 2020.

[15] A. Zareian, S. Karaman, and S.-F. Chang, "Bridging knowledge graphs to generate scene graphs," in *European conference on computer vision*, pp. 606–623, Springer, 2020.

[16] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, no. 4, pp. 13–es, 2006.

[17] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[18] M. Essam, D. Khattab, H. A. Shedeed, and M. F. Tolba, "An enhanced object detection model for scene graph generation," in *International Conference on Advanced Intelligent Systems and Informatics*, pp. 333–343, Springer, 2023.

[19] J. Gu, H. Zhao, Z. Lin, S. Li, J. Cai, and M. Ling, "Scene graph generation with external knowledge and image reconstruction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1969–1978, 2019.

[20] B. Dai, Y. Zhang, and D. Lin, "Detecting visual relationships with deep relational networks," in *Proceedings of the IEEE conference on computer vision and Pattern recognition*, pp. 3076–3086, 2017.

[21] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," in *European conference on computer vision*, pp. 852–869, Springer, 2016.

[22] Y. Li, W. Ouyang, X. Wang, and X. Tang, "Vip-cnn: Visual phrase guided convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1347–1356, 2017.

[23] Y. Li, W. Ouyang, B. Zhou, J. Shi, C. Zhang, and X. Wang, "Factorizable net: an efficient subgraph-based framework for scene graph generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 335–351, 2018.

[24] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5410–5419, 2017.

[25] Y. Cong, M. Y. Yang, and B. Rosenhahn, "Reltr: Relation transformer for scene graph generation," *arXiv preprint arXiv:2201.11460*, 2022.

[26] E. Adamopoulou and L. Moussiades, "An overview of chatbot technology," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 373–383, Springer, 2020.

[27] A. Azaria, "Chatgpt usage and limitations," 2022.

[28] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, no. 4, pp. 681–694, 2020.

[29] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[30] S. Wiseman, S. M. Shieber, and A. M. Rush, "Challenges in data-to-document generation," *arXiv preprint arXiv:1707.08052*, 2017.

[31] D. Deng, Y. Jiang, G. Li, J. Li, and C. Yu, "Scalable column concept determination for web tables using large knowledge bases," *Proceedings of the VLDB Endowment*, vol. 6, no. 13, pp. 1606–1617, 2013.

[32] C. Rebuffel, L. Soulier, G. Scoutheeten, and P. Gallinari, "A hierarchical model for data-to-text generation," in *European Conference on Information Retrieval*, pp. 65–80, Springer, 2020.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[34] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[35] S. Chaudhuri, K. Ellis, O. Polozov, R. Singh, A. Solar-Lezama, Y. Yue, *et al.*, "Neurosymbolic programming," *Foundations and Trends® in Programming Languages*, vol. 7, no. 3, pp. 158–243, 2021.

[36] S. Harris, A. Seaborne, and E. Prud'hommeaux, "Sparql 1.1 query language," 2013. `https://www.w3.org/TR/sparql11-query/` [Accessed on 21.01.2023].

[37] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, "Cypher: An evolving query language for property graphs," in *Proceedings of the 2018 International Conference on Management of Data*, pp. 1433–1445, 2018.

[38] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[39] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," 2016.

[40] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," *arXiv preprint arXiv:1806.03822*, 2018.

[41] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

9 Appendices

A Evaluation example *Desk*

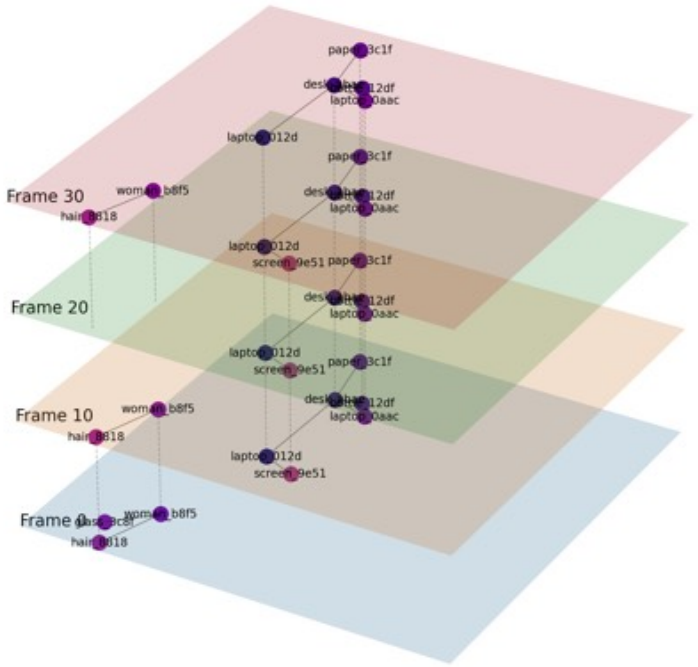

Figure 7: Temporal graph visualization of the desk example. Only every tenth frame is represented for better legibility.

Listing 2 Output of the symbolic link for the first three frames.

```
Creating frame graph...
-------------
Inserting framegraph with id 0
-------------
Creating new node laptop_012d for laptop and match confidence -1
Creating new node desk_abae for desk and match confidence -1
Creating new node bottle_12df for bottle and match confidence -1
Creating new node paper_3c1f for paper and match confidence -1
Creating new node laptop_0aac for laptop and match confidence -1
Creating new node woman_b8f5 for woman and match confidence -1
Creating new node glass_3c8f for glass and match confidence -1
Creating new node hair_8818 for hair and match confidence -1
Creating new node screen_9e51 for screen and match confidence -1
Creating new edge: laptop on desk
Creating new edge: bottle on desk
Creating new edge: paper on desk
Creating new edge: laptop on desk
Creating new edge: woman wearing glass
Creating new edge: woman has hair
Creating new edge: screen on laptop
Creating frame graph...
-------------
Inserting framegraph with id 1
-------------
Updating node laptop_012d with laptop and match confidence 0.9186443657820065
Updating node desk_abae with desk and match confidence 0.8430362907637445
Updating node bottle_12df with bottle and match confidence 0.9678770093269701
Updating node paper_3c1f with paper and match confidence 0.9774138708093284
Updating node laptop_0aac with laptop and match confidence 0.9755049166296647
Updating node woman_b8f5 with woman and match confidence 0.9051002700279724
Updating node hair_8818 with hair and match confidence 0.8437482709001487
Updating node screen_9e51 with screen and match confidence 0.987209351118191
Updating edge: laptop on desk
Updating edge: bottle on desk
Updating edge: paper on desk
Updating edge: laptop on desk
Updating edge: woman has hair
Updating edge: screen on laptop
Creating frame graph...
-------------
Inserting framegraph with id 2
-------------
Updating node laptop_012d with laptop and match confidence 0.9150998534152743
Updating node desk_abae with desk and match confidence 0.8399614301381382
Updating node bottle_12df with bottle and match confidence 0.9865459655092136
Updating node paper_3c1f with paper and match confidence 0.9936354500360415
Updating node woman_b8f5 with woman and match confidence 0.913035020991817
Updating node hair_8818 with hair and match confidence 0.8447640023865288
Updating node screen_9e51 with screen and match confidence 0.9889406737603946
Creating new node bottle_2360 for bottle and match confidence 0.15
Updating node laptop_0aac with laptop and match confidence 0.9690787072777846
Updating edge: laptop on desk
Updating edge: bottle on desk
Updating edge: paper on desk
Updating edge: woman has hair
Updating edge: screen on laptop
Creating new edge: bottle on desk
Updating edge: laptop on desk
```