Where were my keys? - Aggregating Spatial-Temporal Instances of Objects for Efficient Retrieval over Long Periods of Time

Ifrah Idrees, Zahid Hasan, Steven P. Reiss, and Stefanie Tellex
Dept. of Computer Science, Brown University, Providence, RI

Abstract

Robots equipped with situational awareness can help humans efficiently find their lost objects by leveraging spatial and temporal structure. Existing approaches to video and image retrieval do not take into account the unique constraints imposed by a moving camera with partial view of the environment. We present a **D**etection-based **3**-level hierarchical **A**ssociation approach, **D3A**, to create an efficient query-able spatial-temporal representation of unique object instances in an environment. D3A performs online incremental and hierarchical learning to identify keyframes that best represent the unique objects in the environment. These keyframes are learned based on both spatial and temporal features and once identified their corresponding spatial-temporal information is organized in a key-value database. D3A allows for a variety of query patterns such as querying for objects with/without the following: 1) specific attributes, 2) spatial relationships with other objects, and 3) time slices. For a given set of 150 queries, D3A returns a small set of candidate keyframes (which occupy only 0.17% of the total sensory data) with 81.98% mean accuracy in 11.7 ms. This is 47x faster and 33% more accurate than a baseline that naively stores the object matches (detections) in the database without associating spatial-temporal information.

INTRODUCTION

Home-service robots have great potential to assist human users by retrieving spatial-temporal information[1] about objects from their long-term observations. For example, a person can ask a simple query such as *"where did I leave my keys?"* Service robots with such an ability will be well suited to help the elderly, especially those that have dementia.

There are two lines of works in image and video object retrieval: a) *fixed-view cameras* (Philbin et al. 2007; Yadav and Curry 2019; Kang, Bailis, and Zaharia 2019) which assume full observability of the environment and do not deal with a moving camera (robot), and b) *robotic object retrieval* (Ambruş et al. 2014; Bore, Jensfelt, and Folkesson 2015) that either work on short-term time horizons (do not condense partial view detections of unique object instances for memory or speed efficiency) or perform long-term robotic object retrieval by specifically focusing on point cloud matching without incorporating object's spatial information in the physical environment. These approaches for object retrieval will leave the robot searching over countless detections in visual sensor data from many different time slices, which will take up a lot of space and time. In addition, these detections will contain partial views of different object instances, not all of these objects and their partial views will be relevant to the query. An algorithm that condenses the partial view detections of each unique object instance into a compact and query-able spatial-temporal representation will enable the robot to answer queries about unique object instances more efficiently. To our knowledge, there has not been a single work that addresses all of the following challenges: 1) aggregating detections of object instances from different views in a map over hours of sensory information, 2) handling the object going out of the

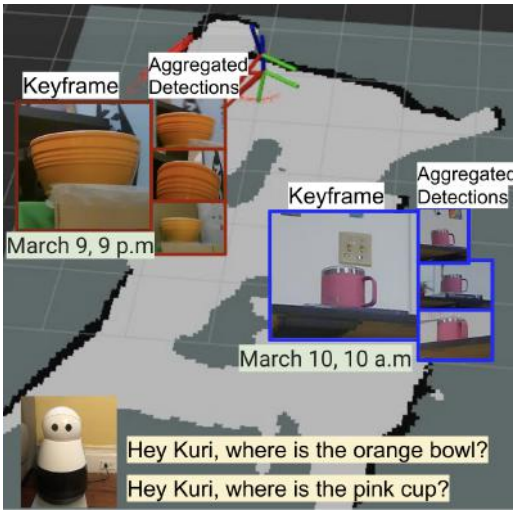

Figure 1: The image shows the retrieved keyframes and associated temporal information for two sample queries answered by D3A while deployed on Kuri robot. The keyframes are overlaid on the map using the spatial information returned by our spatial-temporal representation. Also shown are the partial view detections that are aggregated into the keyframe cluster.

[1]Spatial-temporal information of object refers to the whereabouts of the object such as where the object has been identified in the physical environment of the robot and at what times

view, and 3) accounting for uncertainty in object poses due to noise from the robot pose. To mitigate these challenges, we introduce a new algorithm, D3A that extends the concept of keyframe extraction that has been previously used for video frame retrieval to condense multiview detections of unique objects over both physical space and time for long observation periods.

D3A extracts both spatial-temporal features of object instances in the environment as they get seen by the robot. D3A associates these partial view features with what it has previously seen over the map to condense them. It does so by performing three-tier online, incremental clustering and filtering of unique spatial-temporal instances. While performing association, D3A keeps track of the keyframe that best represents each unique spatial-temporal location of the object. The information about keyframe-centroid clusters are then spatial-temporally indexed in a key-value database, leading to a compact and query-able representation for efficient object retrieval.

This representation allows D3A to identify and return a small set of keyframes for the object(s) in a given query, which enables a person to quickly find the relevant information. D3A is storage efficient and is able to handle the case where the attributes of object instances (color, shape, patterns) asked for in the queries are not known in advance. In prior work all or subset of these attributes were provided at the time of query.

Our algorithm is deployed on a social robot Kuri from Mayfield (2018) that patrolled a robotics lab environment and collected 22 hours of observations over four days[2]. When tested on a set of 150 queries about different objects, D3A returns a small set of keyframes and their spatial-temporal information (which occupy only 0.17% of the total sensory data) and is 47x faster than the baseline model which treats all partial-view detections as independent instances. D3A is also 33% more accurate than the baseline with an accuracy value of 81.98%.

Related Work

In the past, **Keyframe Extraction** techniques have been used to extract the summary key frames to represent video sequences (Wang et al. 2012). A lot of different techniques exists for keyframe extraction such as clustering (Vázquez-Martín and Bandera 2013), energy minimization (Essa, Sidike, and Asari 2015), or online techniques (Elahi and Yang 2020). D3A extends the clustering keyframe extraction technique in videos to incrementally find keyframes that best represent the different views of the unique objects in the environment over time, as the robot continues to capture new video frames.

A lot of work has also been done in **Object Detection** (He et al. 2017; Lin et al. 2017; Girshick 2015; Zakharov, Shugurov, and Ilic 2019) for detecting and re-identifying objects. D3A is built upon the notion of object detection in a three-tier mechanism to match objects and condense their detections over long periods of observations. Additionally, there are **Object Tracking** algorithms that track object instances over time (He et al. 2018; Yilmaz, Javed, and Shah 2006). However, they usually assume input videos with fixed angles and cannot track/associate objects under occlusion or after the objects go out of view. These algorithms are also not useful for retrieval since they do not focus on making the objects' trajectories query-able.

Although many **Image or Video Retrieval** systems have been introduced in the multimedia community, they focus on retrieval at the level of object classes and not instances. These works can be categorized into the following types:
1) Fixed-angle Video Input (Philbin et al. 2007; Yadav and Curry 2019; Kang, Bailis, and Zaharia 2019): These works do not account for the objects' spatial information in a global map. In contrast, D3A can perform object instance association in a partially observable environment monitored by a robot.
2) Query Optimized Object Retrieval (Kang, Bailis, and Zaharia 2019): This work uses information from the query to optimize the retrieval pipeline by searching over all the scaled down image frames in the original dataset. They do not reduce redundancy in the data, whereas D3A focuses on only storing the keyframe information and optimizing the spatial-temporal representation.

There are also works that can handle partial views of objects and perform view-invariant object detection at the category level (Hsieh et al. 2018; Sivic and Zisserman 2003). Hsieh et al. (2018) proposed work similar to ours that can handle many hours of videos captured by a moving camera (that is not a robot) but does not use object locations. We perform an ablation study with this work and show that by using the objects' locations in the map, D3A improves 1) detection of unique object instances and 2) compactness and retrieval performance from our spatial-temporal database. Sivic and Zisserman (2003) assume to have seen a part of the scene captured by different static camera viewpoints to learn the visual vocabulary (which incorporates spatial information) to associate objects seen in later scenes. We perform online, incremental object matching, and do not pre-define the number of clusters/objects in the environment.

There are also works that specifically focus on long-term robotic object retrieval that handle partial observability in visual sensor data (Ambruş et al. 2014; Bore, Jensfelt, and Folkesson 2015). However, these works assume a point cloud of objects as input and introduce matching algorithms specifically in 3D space. D3A, in contrast, does matching using the RGB feature space and limited depth information (information provided by the depth sensors for obstacle detection) that can be derived from the 3D space. These works also do not deal with the objects' spatial information in the environment. Furthermore, they do not focus on making a compact representation of the sensory data for retrieval, but instead store and operate over the original data and use the object attribute "shape" provided in query to optimize retrieval.

[2] A visualization recording of the initial state of the environment, along with its 2D map and sample of the collected dataset can be found at this link: https://github.com/IfrahIdrees/D3A.git

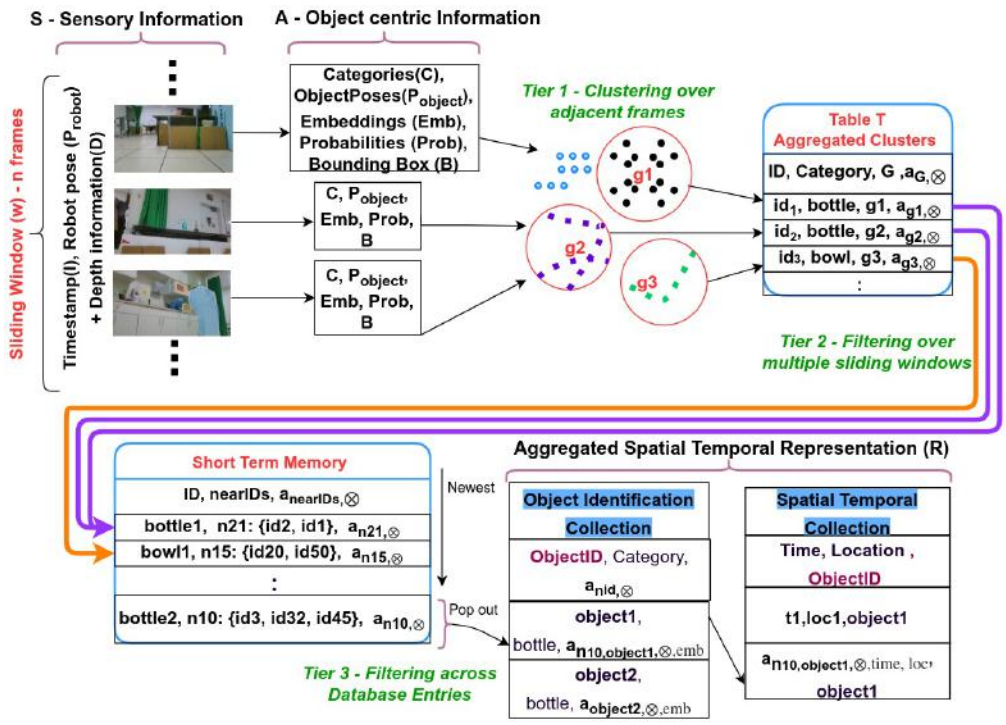

Figure 2: Visualization of D3A Algorithm

Problem Formulation

A robot monitoring the environment over long periods is given a spatial-temporal query Q to retrieve spatial temporal information of target objects seen in the environment. The query can be of any type as listed in Table-1. The robot is equipped with the map of the environment and at any given time step i can gather the following sensory information $s_i =< p_{robot,i}, d_i, f_i >$, where $p_{robot,i} \in P_{robot}$ is the associated robot's pose, and $d_i \in D$ is the corresponding depth information for image frame f_i. The robot collects large amounts of sensor data $S = \{s_1, s_2, ..., s_i\}$ and uses an object detector to extract the relevant object-centric information a_i from every frame f_i. However, the robot has partial observability of the environment and the information required to answer the object retrieval query is going to be dispersed throughout S in the form of multiple partial view detections. To answer object-retrieval queries efficiently, the robot needs to condense the multi-view detections of the objects into a memory efficient and query-able representation. This representation is then used by the robot to return a small set of keyframes, which enables a person to quickly find the relevant information about the target object.

Approach

For efficient spatial-temporal object retrieval in a partially observable environment, we need to compress this data S into a compact and query-able spatial-temporal representation R. For this we first pre-process every s_i to extract the relevant object-centric information a_i, then apply a three-tier online clustering algorithm to aggregate a_i's. The output of D3A is a set of clusters of unique object instances that can be aggregated into the previous time step's compact representation R_{i-1} to create a new representation R_i of the environment for time step i. A diagram of our algorithm creating R is shown in Fig-2. Details of algorithmic implementation can be found in the Appendix.

Pre-processing each frame:

For compactness and speed efficiency, D3A needs to reduce the dimensionality of the visual sensory data f_i in $s_i =< p_{robot,i}, d_i, f_i >$, as well as aggregate multiple detections of the same object over long periods of time. D3A achieves this by extracting minimal amounts of relevant object-centric information a_i:

$$a_i = [< c_{i1}, prob_{i1}, b_{i1}, p_{robot,i1}, emb_{i1} >, < c_{i2}, prob_{i2}, b_{i2}, p_{robot,i2}, emb_{i2} >, ..., < c_{ij}, prob_{ij}, b_{ij}, p_{robot,ij}, emb_{ij} >]$$

for every frame f_i, $c_{ij} \in C_i$, is the category detected for the j^{th} object instance detection in i^{th} image, $prob_{ij}$ is the probability of j^{th} object being assigned the category c_{ij}, and b_{ij} and emb_{ij} is the bounding box and object embedding for the j^{th} detection in f_i. emb_{ij} is used to relate it with the other objects over time and create the compact representation R.

Tier 1- Clustering across adjacent frames

The extracted meta-information for the jth detection in the ith frame – a_{ij} is noisy because of the uncertainty in the robot's pose estimate $p_{robot,i}$ and noise in the extracted object-instance centric information – c_{ij}, $prob_{ij}$, emb_{ij}. To aggregate the different view detections of the

same object and to identify the unique objects in the environment, we consider a sliding window w over sensory data $\{s_{i-n}, ..., s_i\}$. Object-centric information a_{ij} of all detections in all the frames of this sliding window are then clustered to create a set of clusters G, each identifying a unique object instance. The motivation behind this is that the object instances of same category with similar embeddings and positions in space will be associated to the same cluster hence pruning the noisy detections. We assign a unique id $instance_{id} \in ID$ to each cluster $g \in G$ and perform an aggregation operation $\bigotimes$ on every cluster $g \in G$ to create a unified feature representation of each of the cluster. This results in an aggregated representation $a_{g,\bigotimes}$ which is then inserted into a collection T indexed by the unique id $instance_{id}$ of each cluster where ID is the set of all $instance_{id}$'s .

Tier 2 - Filtering across multiple sliding windows

We also need to aggregate the detections of unique instances across the objects' multiple partial views in the past. This requires sharing information across multiple sliding windows via a second-tier filtering. To facilitate this, we maintain a fixed-length short-term memory (STM) indexed over ID and aggregate information in it for all instance id's in T. For an object represented by $instance_{id}$ in collection T, we add a new entry to STM only if the object is a new unique object. Otherwise, if the object has previously been seen by the robot, an entry for this object should already exist in STM and should be updated. We do this by finding a set of $nearIDs$ in STM that are closest in embedding space in cosine similarity and measuring the euclidean distance between them and $instance_{id}$ in physical space. There are three cases to be considered here that symbolize whether the object is new or not and whether it has moved or not: **1)** If $nearIDs$ set is non-empty and euclidean distance is less than a threshold d_{thresh} then we perform another aggregation operation $\bigodot$ on $instance_{id}$ and the closest $n_{id} \in nearIDs$ to create a unified feature representation for n_{id}. This results in an aggregated representation $a_{n_{id},\bigodot}$ that is updated in STM. **2)** If $nearIDs$ set is non-empty and euclidean distance is greater than a threshold d_{thresh}, we make a new entry in the STM with instance id $instance_{id}$. This new entry with the same $instance_{id}$ symbolizes dynamic objects (where the same object has moved in the environment). **3)** If $nearIDs$ is empty for $instance_{id}$, we also add a new entry in the STM with instance id $instance_{id}$. This represents the scenario where the object has not been seen before. After the aggregation operation, the keyframe of n_{id} is chosen to be the frame with the highest detection probability $prob_{ij}$ among the frames $f_i's$ associate with $instance_{id}$ and n_{id}. If a new instance is to be added to the STM when it has reached its maximum capacity, our algorithm evicts the last recently viewed (least-recently viewed entry), lrv_entry, from the STM and moves it to the persistent storage key-value store R. This eviction strategy ensures that objects currently being viewed by the robot remain in the STM for further noise filtering and aggregation of metadata from partial views.

Type	Language Example	Formal Representation
Q1	Did you ever see an orange bowl?	$orange(x), bowl(y) \mid OIc.find(x, y)$
Q2	When were the cup and bowl seen together?	$bowl(x), cup(y) \mid STc.find(OIc.find(x, y), together)$
Q3	Where did you see the pink cup between 1:00pm and 4:00pm on 1st Jan, 2020?	$pink(x), cup(y), time(t) \mid STc.find(OIc.find(x, y), 1600 >= t >= 1300)$

Table 1: Query Types, Language Examples & Formal Representations

Tier 3- Filtering across Database Entries

The filtered entry lrv_entry from the STM could be directly added as the final aggregated cluster into our compact representation R. However, we want to aggregate detections of unique object instances over not just recent times but also long periods of time. To do so, every time a lrv_entry is evicted from STM, our third-tier filtering level looks up and updates the aggregated clusters stored in R. We organize our aggregated spatial-temporal representation R in a key-value database over two collections. The **Object Identification Collection** (OIc) is updated using the object information of the evicted clusters represented by the lrv_entry, while the other store **Spatial-temporal Collection** (STc) stores the object entries indexed over time and position in physical space. Every object inserted in OIc is associated with a unique identifier $ObjectID$ which is used to index STc to get the object detections over space and time.

To update R, we find the set $Records \in OIc$ that are the closest in feature space with lrv_entry in cosine similarity. A new document is inserted into STc if the normalized Euclidean distance between last evicted entry from STM lrv_entry and $Records \in STc$ is greater than d_{thresh}, otherwise the object information in the closest document $record \in Records$ is aggregated, via an operation $\oplus$, applied on the lrv_entry and the document $record$ hence updating the object information in OIc using this aggregated output. This new document insertion in STc symbolizes a moving object while the update symbolizes operation for a static object. For every new entry in OIc, an entry is added in the STc against the $ObjectID$ containing the time and physical location of the new entry.

Query Processing and Answering

Template-based NLP methods (Li 2017) or deep learning (Wang et al. 2019) methods can be used to map a language query to a set X of $ObjectID's$, time intervals, and location areas, which D3A takes as input for object retrieval. Table 1 shows the types of queries, their natural language examples, and formal representations used in our evaluation. This formal representation is used to search over indexed OIc and STc collections of R. The representation R built through our method can handle a wide variety of queries, including: querying for objects with/without the following: 1) specific attributes such as "orange bowl," 2) spatial relationships with other objects such as "cup and bowl together," and 3) time slices which can be either a point in time or a time

Duration	22hrs	3hrs
Total Number of Frames	10,132	991
Frame Rate (per minute)	7.67	7.67
Sensor Data Size (GB)	10.53	1.43
Total Number of Detections	13,565	2,558
Ground Truth Objects	N/A	59
		Static = 49, Dynamic = 10
Number of Times	N/A	Mean = 2.1 $\pm$ 1.3,
Dynamic Objects move		Max = 5.0

Table 2: Dataset Details

interval. Our representation can also efficiently handle negative queries where the answer does not exist in the dataset.

Evaluation

The aim of our evaluation is to test the hypothesis that a database-backed system with our algorithm D3A improves both 1) the compactness of the spatial-temporal representation of objects in the environment and 2) the object retrieval performance by returning a small subset of keyframes as measured by the mean reciprocal rank, miss rate and retrieval time as described in Experiment Design subsection. As a result, the user will have to search over just a few returned results to find the answer to their question. To test our hypothesis, we perform a real-world evaluation, using both a range of synthetic detectors built from ground truth data to control the noise level and real detectors deployed on a mobile robot to assess the system's end-to-end accuracy.

Dataset Collection

The robotics lab environment in which our mobile robot Kuri (Mayfield 2018) patrolled for four days was uncontrolled and cluttered: people were allowed to use the space as is. For consistency, we kept the illumination same throughout the data collection (it was captured in a completely lit lab). The robotics lab area used for data collection and experimentation included the kitchen area and the general area with tables and chairs. This real-world dataset[3] has 10,132 image frames over 22 hours and contains both static and dynamic objects such as various cup and bottles changing there location. The dataset details can be found in Table 2.

Dataset/Object Annotation

For evaluation purposes, the collected dataset was manually annotated by the authors. Every unique instance in the dataset was assigned a unique id and a ground truth location that was used to perform object association over time and space. Due to time and manual labour constraints, we were only able to annotate 3 hours of data.

Parameter Selection

The Euclidean distance threshold d_{thresh} , sliding window length, normalized threshold for feature matching (cosine similarity), and size of short-term memory (STM) were set to 0.5m, 10, 0.4, and 400 (our RAM's maximum capacity), respectively. We performed a parameter sweep offline on the collected data and found this setting to be optimal. Increasing the feature matching and distance threshold increased the number of false positives during tier-1 processing. For the STM, the greater the buffer size, the more noise there will be in the object's pose and embedding to be filtered and processed.

[3]A visualization recording of the initial state of the environment, along with its 2D map and sample raw sensory data can be found at https://github.com/IfrahIdrees/D3A.git.

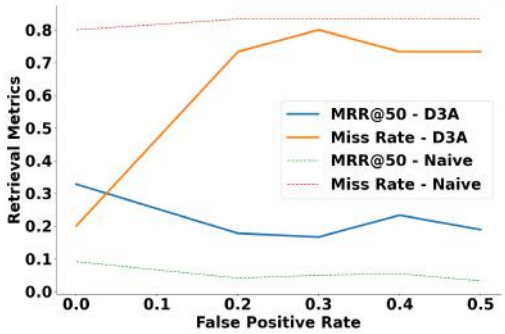

Figure 3: Exp 1 - Retrieval performance with a synthetic embedding generator with increasing uncertainty

Experiment Designs

We conduct four experiments with two baselines. One experiment is a running example comparison.

Baselines: Our first baseline is a "Naive" method that inserts every detection's spatial-temporal information directly into the database without aggregation. This baseline ablates the three-tier processing that is integral to our system for aggregating partial views of the objects, but can still index detections on space and time. We also conduct an ablation of the spatial information from the features to make a "Non-spatial" baseline, similar to Hsieh et al. (2018) which does not utilize spatial information during clustering, to compare the clusters resulting from both methods.

Metric Definitions: We measure the quality of keyframes returned for a query with the MRR@50 evaluation metric, which is the multiplicative inverse of the rank of the correct keyframe in the top 50 returned frames ordered by their probability of correctness. We want the rank of the correct frame to be as close as the start of the list of returned frames and hence higher MRR closer to 1 is better. We measure the miss rate (the number of objects missed by the algorithm during the clustering phase due to detection or classification error, hence cannot be found in the returned frames) to get an idea of our accuracy. We also report the speed efficiency of our system by measuring the mean number of frames and object id's returned for the queries, total time taken by a query divided into retrieval time and evaluation time to find the correct frame by matching against the ground truth.

Exp 1 - Retrieval Performance with a Synthetic Embedding Generator with Increasing Uncertainty The performance of D3A depends on the accuracy of the object detector and object pose estimates. In this experiment, we design a synthetic embedding generator that replaces the object detector and color histogram. Our synthetic embedding

	Type	Query Precision	Aggregation Method	Miss Rate↓	Mean Number of Frames ↓	Mean Number of Object IDs ↓	MRR @50 ↑	Mean Retrieval Time (ms) ↓	Total Evaluation Time (ms) ↓
1			Naive	0.00	1.00	1.00	1.00	0.02	10.30
2		Perfect	D3A+P	0.00	1.00	1.00	1.00	0.02	3.93
3			D3A+D	0.00	10.50	1.00	1.00	0.02	6.63
4			Naive	0.33	792.73	339.00	0.19	22.24	2,567.62
5	Q1	Category	D3A+P	0.20	6.29	4.50	0.33	0.96	101.31
6			D3A+D	0.20	12.43	1.50	0.29	0.94	104.95
7			Naive	0.93	2,558.00	2,558.00	0.01	119.74	926.67
8		Any object	D3A+P	0.20	38.00	33.00	0.24	0.82	116.15
9			D3A+D	0.13	75.00	25.00	0.20	1.12	89.19
10			Naive	0.00	1.00	1.00	1.00	0.02	2.55
11		Perfect	D3A+P	0.00	1.00	1.00	1.00	0.02	6.94
12			D3A+D	0.00	10.20	1.00	1.00	0.03	23.23
13			Naive	0.88	861.00	861.00	0.04	25.99	2,108.60
14	Q2	Category	D3A+P	0.31	7.50	7.00	0.31	1.63	55.81
15			D3A+D	0.38	18.00	3.50	0.18	1.54	73.26
16			Naive	0.94	2,558.00	2,558.00	0.01	117.96	996.27
17		Any object	D3A+P	0.25	38.00	33.00	0.13	0.85	173.65
18			D3A+D	0.38	75.00	25.00	0.10	1.14	153.19
19			Naive	0.00	1.00	1.00	1.00	0.02	5.05
20		Perfect	D3A+P	0.00	1.00	1.00	1.00	0.02	4.15
21			D3A+D	0.00	5.50	1.00	1.00	0.02	3.53
22			Naive	0.67	96.00	96.00	0.25	336.60	110.58
23	Q3	Category	D3A+P	0.27	1.50	1.50	0.36	1.60	9.20
24			D3A+D	0.47	1.00	1.00	0.33	1.35	5.68
25			Naive	0.86	684.00	684.00	0.08	2,723.35	160.27
26		Any object	D3A+P	0.14	14.00	14.00	0.21	13.25	31.45
27			D3A+D	0.07	12.00	12.00	0.26	11.51	28.93

Table 3: Exp 2 - Retrieval Performance for the three types of queries. The highlighted row corresponding to D3A+D shows performance of our deployed D3A with real detector. D3A+D's performance is better than the Naive Baseline Aggregation method and sightly worse than D3A+P (D3A's implementation with a perfect object detector) in all the query types. ↑ means higher is better, ↓ means lower is better.

generator is parameterized with a false positive rate (fpr)[4] and false negative rate (fnr)[5]. We keep the fnr low since we assume that the object is not in the environment if the detector fails to detects the object. We annotate ground truth object embeddings as one-hot encodings ohe of the unique instances, where each index represents the unique id of a ground truth object. We then use the fpr and fnr to flip the bits of the ohe and return the transformed encoding as the synthetic embedding generator's output. **Results**: We tested our system on Q1 type - *Did you ever see an orange bowl?* for all ground truth object instances as mentioned in Table 1. Results are shown in Fig-3. Our method's MRR@50 decreases as fpr increases. Despite this trend, D3A's MRR@50 remains considerably higher than Naive's MRR@50. This indicates that even when the object detector used in D3A is not good, the user will always find the correct object's frame earlier with D3A than with the Naive baseline. D3A's Miss Rate increases as the synthetic detector becomes noisier, but is always lower than Naive's Miss rate. The slight decreasing trend of D3A's Miss Rate for $fpr > 0.3$ is because with a higher probability of flipping the bits of the ohe, the probability of different object instances being assigned a similar embedding also increases. This causes D3A to consider a higher proportion of object instances to be similar, leading to more cluster formations (because of different locations) and storage of their respective keyframes. At the time of retrieval for a given object, this causes more keyframes to be returned one of which includes the queried object.

[4] fpr denotes the probability of falsely labelling the detected instance

[5] fnr denotes the detector's probability of not detecting the object

Exp 2 - Query Retrieval Performance Our second experiment compares the retrieval performance of two versions of D3A against the Naive Baseline. The first is $D3A + D$ - D3A with a real detector, Dectectron (Girshick et al. 2018), and color histogram and second is $D3A + P$ - D3A with a synthetic perfect sensor. We experimented over the 3 query types in Table 1 and the 59 ground truth object instances annotated in the data. The number of instances per query ranged from 15 -20 depending on the query, making the total number of queries to be 150.

Query Matching: Another factor affecting the retrieval performance in this experiment is the matching of the desired object in the natural language query to an $ObjectID$ in the database. To examine this axis, we discretize the query matching precision into three levels:

- *Perfect*: The query matching component can map an object in the language query to an exact $ObjectID$ in the database, and removes any level of uncertainty.

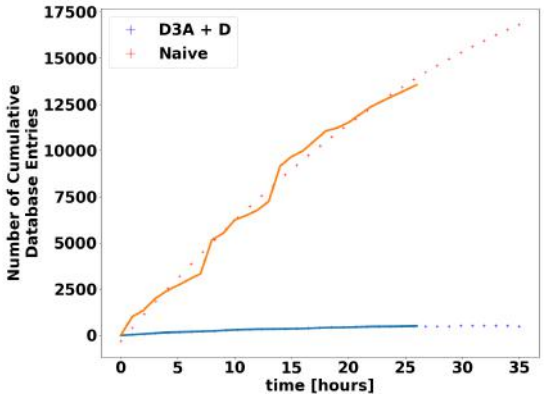

Figure 4: Exp 3 - Cumulative number of insertions to the database per hour

- *Category*: The matching component is noisier and is unable to uniquely identify the queried object (*e.g.,* orange bowl), and instead returns a set of $ObjectID's$ for all objects from the same category (*e.g.,* bowl).
- *Any Object*: The matching component is the noisiest and cannot extract even the category-level information from the language query, and thus maps the query to the most generic set of all the $ObjectID's$ in the database.

Results: The results are shown in Table 3. With *Perfect Query Precision* for all three types of queries Q1 [*rows: 1-3*], Q2 [*rows: 10-12*], and Q3 [*rows: 19-21*], both the mean retrieval time and total evaluation time of $D3A + D$ are less than those of the "Naive" baseline. On average, $D3A+D$'s total (retrieval + evaluation) time is **1.96x** slower than $D3A+P$ and **1.48x** faster than the baseline. This translates to improved user experience as the users' query will be answered more quickly and as expected the miss rate is 0 because of the perfect query matching precision.

With *Category Query Precision* for Q1 [*rows: 4-6*], Q2 [*rows: 13-15*], and Q3 [rows:*22-24*], the mean number of frames returned by $D3A + D$ is **55x** smaller than that of the "Naive" baseline, and only **5 counts** higher than $D3A+P$'s on average. The MRR@50 for $D3A+D$ is **0.26**, which signifies that if the queried object exists in the database, it will on average be found in the fourth frame. An empty set is returned for negative queries. While the baseline on average finds the object in the sixth frame, the probability of it not finding an object of interest in the first 50 frames (average miss rate) is **62.5%**, which is really high compared to $D3A + D$'s **34.7%**. On average, $D3A + D$ took **23ms** to retrieve and evaluate all the frames for all the object instances while the baseline took **500ms**. These evaluation times, however, are just for 3 hours of data. These times will increase linearly with the amount of sensor data for the "Naive" and increase the human users' waiting time for their query to be answered.

The worst retrieval performance is with *Any Object Query Precision* for Q1 [*rows: 7-9*], Q2 [*rows: 16-18*], and Q3 [*25-27*)]. Even then, $D3A + D$ returns **54** frames on average, compared to the **30** frames returned by $D3A+P$ and **1933** frames returned by the "Naive" baseline. This difference is also reflected in their retrieval and evaluation times. The baseline on average finds the queried object in the 25th frame ($MRR = 0.04$), while $D3A + D$ and $D3A + P$ finds the object around the fifth frame. Even with the worst query precision where the queried object is mapped to all the $ObjectID's$ in the database, D3A still achieves better retrieval performance by clustering detections and thus looking through fewer instances.

Performance Metric		**Naive**	**D3A+D**
Database Size (MB)		3312	244
Total Processing Time		0m 8s	2m 26s
Average Query Response Time (ms)		503.59	10.63
Total Number of Unique Object IDs		**2558**	**25**
Number of Duplicates per Object	Mean	44.10 ± 75.66	1.19 ± 0.46
	Max	369.00	3.00
Mean Accuracy (%)		48.85	81.98
Average Number of Keyframes Returned	U	1.00	8.73
	C	**583.24**	**10.47**
	A	**1933**	**54.00**
Mean Reciprocal Rank ($N = 50$)	U	1.00	1.00
	C	**0.16**	**0.26**
	A	0.03	0.19

Table 4: System Performance (On 3 Hours of Data)
* The bolded numbers are highlighted results discussed in the text

D3A's overall accuracy is **81.98%** with a retrieval time that is **47x** faster than the "Naive" baseline. This can lead to improved user experience since with D3A, the user's query will be answered by searching over a smaller number of key frames (by **97%**) than with the baseline.

Exp 3 - Compactness Comparison We measure the cumulative number of object id insertions in the database per hour over the complete 22 hours for both $D3A + D$ and the "Naive" baseline. As seen in Fig-4, the number of insertions per hour for $D3A + D$ is much less than that of the baseline and scales well as the amount of data increases, demonstrating D3A's success in aggregating partial views. D3A takes two orders of magnitude more time to process the raw data than the baseline, but outputs a representation **14.7x** more compact and efficient in answering questions, as shown in Table 4. This demonstrates that D3A allows for scalable spatial-temporal representation of objects.

Exp 4 - Comparison with the "Non-spatial" Baseline

In this experiment, we visually compare the aggregated clusters formed for the unique object instances by $D3A + D$ and the "Non-spatial" baseline. Fig-5 shows the baseline's aggregated detections for the pink cup asked in Fig-1. The baseline clusters all object instances that are similar in embedding space, irrespective of their location in the map. The images' different bounding box colors show the respective clusters they were mapped to by D3A based on both the embedding features and spatial information. The clusters formed by D3A most closely resemble ground truth. Furthermore, the "Non-spatial" baseline will fail to answer

queries of type Q3 in Table 1.

Discussion

Our deployed algorithm $D3A+D$ shows memory and speed efficiency in clustering together partial views of the same object, even when the object has moved to different locations in the environment over time. In such cases, the same object in each of its new location is assigned to a different keyframe centroid, but is associated with the same $ObjectID$. This is shown in Fig-5 where the pink cup is moved from the table in img3 to the kitchen counter in img4 and assigned to two separate keyframe centroid by D3A, but is assigned the same $ObjectID$, symbolized by the same bounding box color.

D3A can also disambiguate unique object instances with similar features. An example is shown in Fig-5 where D3A assigns the two red cups in img1 and img2 to different clusters. However, our algorithm cannot perfectly retrieve all the locations a unique object was at every moment in time. This is because our system sometimes fails to distinguish objects when they have similar embedding features and are close together in Euclidean space (below d_{thresh}). In this case, D3A will assign the same $ObjectID$ and keyframe centroid to two unique object instances. An example of this is shown in Fig-5 where img1 of the red cup is assigned to the cluster for the pink cup since they have similar colors and are close together in the physical world. This causes D3A to miss the associated information of the wrongly assigned red cup and not be able to retrieve this information in response to an object retrieval query related to the red cup.

Furthermore, if an object is only seen in one frame of a sliding window, our algorithm will characterize it as noise to account for errors in the detection. This, however, causes D3A to sometimes miss unique object instances. Overall, D3A recognized 76 unique object locations in the map, but because of the above reasons assigns them to 25 object instances instead of the ground truth 59, while considering 18 of the objects as noise. As shown in Table 4, the average number of duplicated $ObjectID's$ assigned to a ground truth object instance by D3A is 1.19 ± 0.46, which is 36X smaller than that of "Naive." In the case of the "Naive" baseline, every partial view detection of the object is treated as a unique object, leading to a total number of unique detections much greater than the number of ground truth objects. Fig-3 and Table 3 show that increasing the false positive rate of the visual detector and decreasing the query precision degrades object retrieval performance from the spatial-temporal representation created by D3A. Nevertheless, the deployed D3A+D algorithm on Kuri with the real-world detector performs better than the baseline in terms of memory and speed efficiency for object retrieval. D3A+D also performs better than "Naive" despite the fact that they both suffer from drifting localization of the robot's pose over time. This is because D3A's three-tier clustering and filtering structure was able to filter out some of the localization drift as noise in the objects' sensor data.

We took the first step in aggregating the detections of objects over space and time with our proposed algorithm - D3A, and defer the reduction of duplicates by using improved deep-learned object embeddings instead of color histograms to future work. D3A can enable service robots to find misplaced objects for the elderly over long periods of time. Our research also opens venues for object retrieval in other surveillance related applications. Our work has only been evaluated in an environment where the pre-built map was provided to us. It will be interesting to explore how our system can be adopted to work in unknown open domain environments. We plan to conduct a human user study in the future to investigate their experience when querying D3A and searching over the small subset of returned keyframes to find answers to their object retrieval queries.

Figure 5: Exp 4 - Visualization of the keyframe centroid and the aggregated detections by the "Non-spatial" baseline (Hsieh et al. 2018) for the pink cup. All the square images are the detections and the different bounding box colors show the respective clusters they were mapped to by D3A based on both the embedding feature and spatial information. The dotted line connects clusters that were mapped to the same *ObjectID*.

Conclusion

We present a novel algorithm for robots to efficiently answer spatial-temporal queries about objects in the environment over long periods of time. Our algorithm aggregates partial view detections of unique instances to create a compact and query-able representation of the objects. By explicitly performing detection-based three-level association to identify the keyframes for unique object instances, our algorithm significantly outperforms baselines in answering queries in terms of the retrieved frames' accuracy, mean reciprocal rank and retrieval time. A robot deployed with our algorithm was able to process 22 hours of sensor data and develop a compact representation of objects in its environment, which is an encouraging step towards enhancing the sensory capabilities of home-service robots that can help the elderly find their lost, forgotten objects.

Acknowledgments

This work was supported by NSF under grant award IIS-1652561 and an award from Echo Labs. The authors would like to thank Baber Khalid, Thao Nguyen, and Eric Rosen for their valuable feedback and insights on the research and Baber Khalid for being a supportive husband.

References

Ambruş, R.; Bore, N.; Folkesson, J.; and Jensfelt, P. 2014. Meta-rooms: Building and maintaining long term spatial models in a dynamic world. In *2014 IEEE/RSJ IROS*, 1854–1861. IEEE.

Banker, K. 2011. *MongoDB in action*. Manning Publications Co.

Bore, N.; Jensfelt, P.; and Folkesson, J. 2015. Retrieval of arbitrary 3D objects from robot observations. In *2015 European Conference on Mobile Robots (ECMR)*, 1–8. IEEE.

Brás, A.; and Neto, P. 2018. Unsupervised Feature Extraction from RGB-D Data for Object Classification: a Case Study on the YCB Object and Model Set. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, 3673–3678. IEEE.

Elahi, G.; and Yang, Y.-H. 2020. Online Learnable Keyframe Extraction in Videos and its Application with Semantic Word Vector in Action Recognition. *arXiv preprint arXiv:2009.12434* .

Essa, A.; Sidike, P.; and Asari, V. 2015. A modular approach for key-frame selection in wide area surveillance video analysis. In *2015 National Aerospace and Electronics Conference (NAECON)*, 41–44. IEEE.

Girshick, R.; Radosavovic, I.; Gkioxari, G.; Dollár, P.; and He, K. 2018. Detectron. https://github.com/facebookresearch/detectron.

Girshick, R. B. 2015. Fast R-CNN. *CoRR* abs/1504.08083. URL http://arxiv.org/abs/1504.08083.

He, A.; Luo, C.; Tian, X.; and Zeng, W. 2018. A Twofold Siamese Network for Real-Time Object Tracking. *CoRR* abs/1802.08817. URL http://arxiv.org/abs/1802.08817.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B. 2017. Mask R-CNN. *CoRR* abs/1703.06870. URL http://arxiv.org/abs/1703.06870.

Hsieh, K.; Ananthanarayanan, G.; Bodik, P.; Venkataraman, S.; Bahl, P.; Philipose, M.; Gibbons, P. B.; and Mutlu, O. 2018. Focus: Querying large video datasets with low latency and low cost. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 269–286.

Kang, D.; Bailis, P.; and Zaharia, M. 2019. Challenges and Opportunities in DNN-Based Video Analytics: A Demonstration of the BlazeIt Video Query Engine. In *CIDR*.

Khan, K.; Rehman, S. U.; Aziz, K.; Fong, S.; and Sarasvady, S. 2014. DBSCAN: Past, present and future. In *The fifth (ICADIWT 2014)*, 232–238. IEEE.

Li, F. 2017. *Querying RDBMS Using Natural Language*. Ph.D. thesis, University of Michigan.

Lin, T.; Goyal, P.; Girshick, R. B.; He, K.; and Dollár, P. 2017. Focal Loss for Dense Object Detection. *CoRR* abs/1708.02002. URL http://arxiv.org/abs/1708.02002.

Mayfield, R. 2018. Meet Kuri! The Adorable Home Robot. URL https://www.heykuri.com/.

Philbin, J.; Chum, O.; Isard, M.; Sivic, J.; and Zisserman, A. 2007. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE CVPR*, 1–8. ISSN 1063-6919. doi:10.1109/CVPR.2007.383172.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. Y. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, 5. Kobe, Japan.

Sivic, J.; and Zisserman, A. 2003. Video Google: A text retrieval approach to object matching in videos. In *null*, 1470. IEEE.

Spannbauer, A. 2018. Finding and Using Images' Dominant Colors using Python and OpenCV. URL https://adamspannbauer.github.io/2018/03/02/app-icon-dominant-colors/.

Vázquez-Martín, R.; and Bandera, A. 2013. Spatio-temporal feature-based keyframe detection from video shots using spectral clustering. *Pattern Recognition Letters* 34(7): 770–779.

Wang, B.; Shin, R.; Liu, X.; Polozov, O.; and Richardson, M. 2019. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. *CoRR* abs/1911.04942. URL http://arxiv.org/abs/1911.04942.

Wang, M.; Hong, R.; Li, G.; Zha, Z.; Yan, S.; and Chua, T. 2012. Event Driven Web Video Summarization by Tag Localization and Key-Shot Identification. *IEEE Transactions on Multimedia* 14(4): 975–985. ISSN 1520-9210. doi:10.1109/TMM.2012.2185041.

Xu, Y.; Mo, T.; Feng, Q.; Zhong, P.; Lai, M.; Eric, I.; and Chang, C. 2014. Deep learning of feature representation with multiple instance learning for medical image analysis. In *2014 IEEE (ICASSP)*, 1626–1630. IEEE.

Yadav, P.; and Curry, E. 2019. VidCEP: Complex Event Processing Framework to Detect Spatiotemporal Patterns in Video Streams. In *2019 IEEE International Conference on Big Data (Big Data)*, 2513–2522. IEEE.

Yilmaz, A.; Javed, O.; and Shah, M. 2006. Object tracking: A survey. *Acm computing surveys (CSUR)* 38(4): 13–es.

Zakharov, S.; Shugurov, I.; and Ilic, S. 2019. DPOD: Dense 6D Pose Object Detector in RGB images. *CoRR* abs/1902.11020. URL http://arxiv.org/abs/1902.11020.

Appendix

Algorithmic Implementation

We deploy D3A on a mobile robot Kuri (Mayfield 2018) that patrolled a robotics lab environment for four days. Our system performs real-time object detection with Facebook's AI Research software system - Detectron (Girshick et al. 2018) to extract $c_{ij}, prob_{ij}$ for every detection. We use a color histogram (Spannbauer 2018) to approximate an embedding emb_{ij} of each object instance detection. We select the color histogram method due to its simplicity. However, our algorithm is not constraint to using the color histogram as the instance feature extractor. More sophisticated deep learning methods that use RGB or RGB-D images (Brás and Neto 2018; Xu et al. 2014) can also be used.

In the subsection Experiment 1 of section Evaluation, we show how varying the type of object detector (changing its false positive rate) affects the results of our algorithm. Notably, our approach is also not constrained to any particular number of objects, object size or shape, or object dynamicness (*i.e.,* the objects' locations can change over time). We are only constrained by the object categories the detector was trained to detect. Processing of the visual data and pose estimates was done off-board on a stand-alone machine. The PC setup included an Intel(R) Core(TM) i7-6700 CPU (3.40GHz x 8), and a GeForce GTX 1070/PCIe/SSE2 GPU running Ubuntu 16.04. D3A was run online while the PC communicated with the Kuri robot over ROS (Quigley et al. 2009). For Tier 1-clustering of detections across adjacent frames we use density-based spatial clustering (DBScan) (Khan et al. 2014), a method known to be good at handling outliers/noise within the dataset. While other clustering methods can also be used, we defer the analysis of their impact on D3A's performance for future work.

All the aggregation operations $\odot$, $\oplus$ and $\otimes$ that we applied are the weighted average functions. These aggregation functions can also be different. Weighted average was selected because of its simplicity and reasonable empirical performance for object retrieval on our smaller dataset of < 3 hours of data. More sophisticated methods can also be used in place of the weighted average. However, as shown in section – Experiment Designs of Evaluation, with just the weighted average, D3A still performs better than the baseline.

We organized our aggregated spatial-temporal representation R over two collections in *MongoDB* (Banker 2011), a database that supports large volumes of both data and traffic. The two document Mongo-DB collections **Object Identification Collection** (OIc), and **Spatial-temporal Collection** (STc) are as described in Section – Tier 3- Filtering across database entries of Approach. OIc in our implementation, stores the object information using the associate embedding of the cluster, the category assigned to the cluster and a weight which encodes the frequency of detections for this object and is used in the aggregation operation between $record$ in OIc and lrv_entry evicted by the STM. For natural language processing of the queries, we annotate the queries (use the template-based method) to convert them into the logical form shown in Table 1. These logical forms are then used to query the collections for relevant spatial-temporal information. In the subsection Experiment 2 of section Evaluation we show how the performance of D3A changes with varying the uncertainty in the processing of the query but still it is able to perform better than the baselines.