Künstl Intell (2013) 27:119–128
DOI 10.1007/s13218-013-0239-2

TECHNICAL CONTRIBUTION

Stream-Based Hierarchical Anchoring

Fredrik Heintz · Jonas Kvarnström · Patrick Doherty

Received: 30 September 2012 / Accepted: 8 March 2013 / Published online: 28 March 2013

Abstract Autonomous systems situated in the real world often need to recognize, track, and reason about various types of physical objects. In order to allow reasoning at a symbolic level, one must create and continuously maintain a correlation between symbols denoting physical objects and sensor data being collected about them, a process called *anchoring*.

In this paper we present a stream-based hierarchical anchoring framework. A classification hierarchy is associated with expressive conditions for hypothesizing the type and identity of an object given streams of temporally tagged sensor data. The anchoring process constructs and maintains a set of *object linkage structures* representing the best possible hypotheses at any time. Each hypothesis can be incrementally generalized or narrowed down as new sensor data arrives. Symbols can be associated with an object at any level of classification, permitting symbolic reasoning on different levels of abstraction. The approach is integrated in the DyKnow knowledge processing middleware and has been applied to an unmanned aerial vehicle traffic monitoring application.

This work is partially supported by grants from the Swedish Foundation for Strategic Research (SSF) project CUAS, the Swedish Research Council (VR) Linnaeus Center CADICS, the ELLIIT Excellence Center at Linköping-Lund in Information Technology, the Vinnova NFFP5 Swedish National Aviation Engineering Research Program, and the Center for Industrial Information Technology CENIIT.

F. Heintz (✉) · J. Kvarnström · P. Doherty
Department of Computer and Information Science, Linköpings Universitet, 581 83 Linköping, Sweden
e-mail: frehe@ida.liu.se

J. Kvarnström
e-mail: jonkv@ida.liu.se

P. Doherty
e-mail: patdo@ida.liu.se

1 Introduction

The ability to recognize, track, and reason about various types of physical objects is essential for many autonomous systems situated in the real world. One difficult issue to resolve is how to create and maintain a consistent correlation between symbolic representations of these objects and the sensor data that is being continually collected about them, a process called *anchoring* [3].

As a motivating example, suppose a human operator aims to maintain situational awareness about traffic in an area using static and mobile sensors such as surveillance cameras and unmanned helicopters. Reducing the amount of information sent to the operator reduces her cognitive load, helping her to focus her attention on salient events. Therefore each sensor platform should monitor traffic situations and only report back relevant high-level events, such as reckless overtakes and probable drunk driving. To detect such events it is not sufficient that we can identify those cars and other objects that occur in a single image. Instead each object must have a persistent identity over time, so that we can reliably detect complex events such as a particular car being behind another car, then beside it, and finally in front of it. To achieve this the symbol representing each object must be properly anchored.

Tracking an object through a series of images is an established problem, and there are many effective solutions for the case where the object can be tracked without interruptions. However, we must also consider the case where an object is temporarily hidden by obstacles (or tunnels in the case of traffic), and where many similar objects are present in the world. In this case pure image-based tracking does not provide a complete solution: The information available in the image itself is generally not sufficient to infer correct object identities over longer periods of time. Instead we must include additional knowledge about the world at higher abstraction levels, such as the normative characteristics of specific classes of physical objects. In the case of traffic, this could include the layout of the road network as well as the typical size, speed, and driving behavior of cars. This is a central problem, and in fact, it has been argued that anchoring is an extension to classical tracking approaches which handles missing data in a principled manner [7].

In this article we present a stream-based anchoring framework and a realization that extends the existing knowledge processing middleware framework DyKnow [10, 14]. The solution uses *object linkage structures* to incrementally classify and track objects and to anchor these objects to symbolic identifiers on multiple levels of abstraction, allowing a hierarchy of object types. A metric temporal logic is used to declaratively specify temporally extended conditions for classification as well as for reacquiring object identities.

Two important advantages of our approach compared to existing approaches is that it explicitly models time, allowing complex temporal classification conditions to be expressed, and that it supports hierarchical anchoring where an object is incrementally anchored to more and more specific object types, allowing multiple related representations of objects to be created and used concurrently.

2 Anchoring Using Object Linkage Structures

The objective of the anchoring process is to connect symbols to sensor data originating in the physical world, which requires processing on many abstraction levels. At the lowest level, the input consists of raw sensor data which can be processed by techniques adapted to each particular type of sensor. For example, a variety of image processing techniques can be applied to input from a color camera. Even though such techniques lack higher-level knowledge about objects or their behavior, they can nevertheless often provide some information about which parts of the raw sensor input pertain to a single physical object. They may even be able to track such objects and give them persistent identities over shorter periods of time. For example, image processing can extract "blobs" within a frame, each corresponding to a single physical object. Image-based tracking might then track blobs over multiple frames, until losing track due to for example obstacles or changing visual conditions.

Clearly, anchoring should be able to make use of all information such techniques can provide. We therefore represent the sensor input to the anchoring process as a stream of *percepts*, each of which is an object whose attributes change

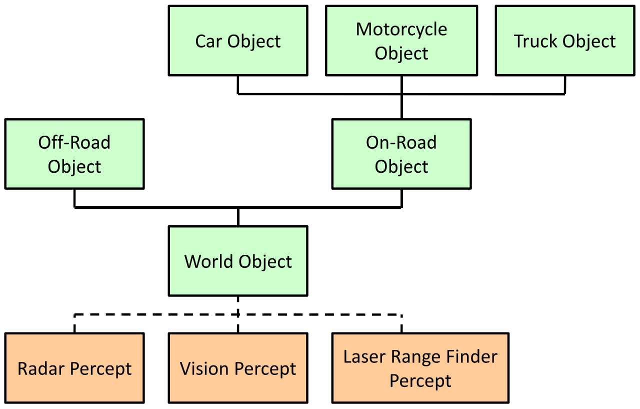


Fig. 1 The example percept/object hierarchy used in the traffic monitoring scenario

but whose identity persists. A *vision percept*, for example, could include color and size information that image processing has extracted for a moving or stationary blob. We may also have *radar percepts* or *laser range finder percepts*, which might or might not originate in the same physical objects. Anchoring then consists of the more difficult task of consistently associating symbolic identities with percepts for specific physical objects over the long term, even when no information arrives about a particular object for extended periods of time. This allows for grounded high-level symbolic reasoning, where attributes of objects may be computed from sensor data even though no sensor completely tracks the object through its lifetime.

Rather than doing anchoring in a single step, as in most current approaches, we define an incremental process of object classification and (re-)identification. This process builds on a hierarchy of percept and object types.

An example hierarchy for the traffic monitoring scenario can be seen in Fig. 1. A *world object* represents a physical object. Its attributes are based on information from one or more linked percepts and include the absolute coordinates of the object in the physical world. World objects can be *on-road objects* moving along roads or *off-road objects* that travel or exist anywhere. An on-road object has attributes representing the road segment or crossing the object occupies, making more qualitative forms of reasoning possible, and an improved position estimation which is snapped to the road. Finally, an on-road object could be a *car*, a *motorcycle*, a *truck*, or neither. Each level in the hierarchy adds more abstract and qualitative information while still maintaining a copy of the attributes of the object it was derived from. Thus, an on-road object contains both the original position from the world object and the position projected onto the road network.

Hypotheses about object types and identities must be able to evolve over time. For example, while it might be determined quickly that a world object is an on-road object, more time may be required to determine that it is in fact a car. Also, what initially appeared to be a car might later turn out to be better classified as a truck. To support incremental addition of information and incremental revision of hypotheses, a single physical object is not represented as an indivisible object structure but as an *object linkage structure*.

An object linkage structure consists of a set of *objects* which are *linked* together (note that percepts are also considered to be objects). Each object has a type and is associated with a symbol, and represents information about a particular physical object at a given level of abstraction. A symbol is *anchored* if its object is part of an object linkage structure that is grounded in at least one percept.

2.1 Incremental Generation of Object Linkage Structures

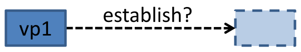
As percepts arrive from low-level sensor processing, object linkage structures must be incrementally generated, updated, and maintained. For example, when a new percept with a previously unseen identity arrives, it may be hypothesized to correspond to a new or existing object at a higher level of abstraction. If a new object is inferred, it may also correspond to an object at an even higher level of abstraction, and so on. Furthermore, when an existing percept or object is updated with new information, this may cause existing classification hypotheses to be invalidated.

The conditions determining when this should happen are written in an expressive temporal logic, similar to the well known Metric Temporal Logic [18]. Such formulas are defined over infinite state sequences but must be tested in real time, and are therefore evaluated incrementally using *progression* (Sect. 3). Informally, given a formula and a new timed state including percept and object attributes, progression determines that the information received up to this point in time is sufficient to verify that the formula is true regardless of future states (the formula "is progressed to true" or "becomes true"), that the formula is false, or that its status still depends on information to arrive in the future.

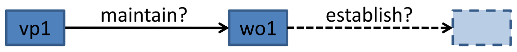
Establishing Links Whenever a new percept or other object of a given type A is created, the anchoring system must consider whether this object can also be classified as belonging to a new object of one of the immediate subtypes. Anchoring therefore immediately instantiates a unary *establish condition* for every immediate subtype B and then begins progressing these conditions. For example, a new world object might be classified as corresponding to a new on-road object if it has been observed on a road for at least 30 seconds, or as a new off-road object if another condition holds.

If and when a state arrives that causes the establish condition from A to some subtype B to be progressed to true, a new object structure of type B is created and a link between the objects is established, thereby generating or extending an object linkage structure. The link between the objects indicates that lower-level information about the first object is used to derive higher-level information about the second object, possibly together with information from other sources. This corresponds to the *Find* functionality suggested by Coradeschi and Saffiotti [3], which takes a symbolic description of an object and aims to anchor it in sensor data.

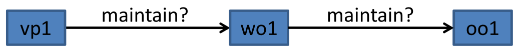
Assume that an image processing system is currently tracking a potential car represented by the vision percept `vp1` and that no other objects have been created. The only parent of *vision percept* in the hierarchy is *world object*, so only such links can be considered. We do not have to consider linking the vision percept to an existing world object, since none exists. Therefore it is sufficient to progress the establish condition from `vp1` to a new world object in order to determine whether a new world object should be generated.



If the establish condition is eventually satisfied (one or more states are received causing the formula to progress to true), a new world object `wo1` is created which is associated with `vp1`. For as long as `wo1` is associated with `vp1`, its state is computed partly or entirely from the state of that vision percept. It is also possible to estimate a model of the behavior of `wo1` using the collected information. This model can later be used to predict the behavior of `wo1` if image processing loses track and the vision percept is no longer updated.

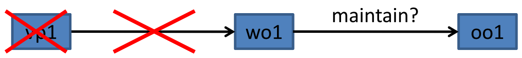


Given a new world object, the establish conditions for *on-road object* and *off-road object* must be progressed. Assume that after a while the establish condition for *on-road object* is progressed to true. Then `wo1` is hypothesized as being a new on-road object, represented by `oo1`, and the object linkage structure is extended to contain three objects.

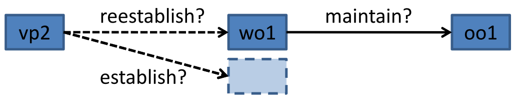


Reestablishing Links The binary *reestablish condition* expresses the condition for an object of type A to be linked to a *known* object of an immediate subtype B, as in the case where a new world object corresponds to an existing on-road object that had temporarily been hidden by a bridge. Whenever a new object of type A is created, the anchoring system immediately instantiates and begins to progress the re-establish condition for every known object of type B that is not already linked to an object of type A. If and when one of these conditions becomes true, a link is created between the associated objects. This corresponds to the *Reacquire* functionality of Coradeschi and Saffiotti [3].

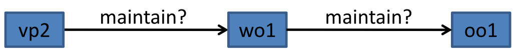
Assume for example that after a while, image processing loses track of the blob represented by vp1, which we previously hypothesized to be an on-road object. Then vp1 is removed. The link from vp1 to wo1 is also removed, but despite this the world object wo1 remains and is still linked to oo1. As long as wo1 is not linked to any vision percept its state is predicted using either a general model of physical objects or an individually adapted model of this particular object. Since wo1 is linked to an on-road object it is also possible to use this hypothesis to further restrict the predicted movements of the object as it can be assumed to only move along roads. This greatly reduces the possible positions of the object and facilitates reestablishing a link to it.



Assume further that the image processing system later recognizes a potential car represented by the new vision percept vp2. Since there exists a known world object, wo1, the anchoring system has to determine whether vp2 is a new world object, the known world object wo1, or not a world object at all. This is done by instantiating and progressing both the establish condition on vp2 and the reestablish condition between vp2 and wo1.



Finally, assume that after a number of states have arrived, the establish condition is progressed to false and the (in this case unique) reestablish condition is progressed to true. Then a new link is created from vp2 to wo1 and the attributes of wo1 can be computed from the attributes of vp2. This also means that oo1 is once again anchored.



Maintaining Links Since observations are uncertain and classification is imperfect, any link is considered a hypothesis and is continually validated through a *maintain condition*. Such conditions can compare the observed behavior of an object with behavior that is normative for its type, and possibly with behavior predicted in other ways. For example, one might state that an on-road object should remain continually on the road, maybe with occasional shorter periods being observed off the road due to sensor error.

If progression determines that the condition is violated, the link is removed. However, it is essential for anchoring that sensor data can be anchored even to symbols/objects for which no percepts have arrived for a period of time. Therefore object structures and their symbols are retained for some time when their associated percepts disappear, as shown above, enabling re-classification and re-identification at a later time. This resembles the *Track* functionality of Coradeschi and Saffiotti [3]. To reduce the computational cost, though, objects which are not likely to be found again may be removed. The criteria for determining that an object is not likely to be found again are specific to each object type and can include conditions such as the time since the object was last observed or anchored.

The state of an object without incoming links, i.e. an unanchored object, is predicted based on a model of how objects of this type normally behave. In future work, we may extend this ability by estimating a specific model for this particular individual using system identification or machine learning techniques on the data collected while tracking it.

Top-Down Identification Object linkage structures can be created bottom-up as shown above, by processing incoming percepts and creating a new symbol for each hypothesized object. A similar approach can be used for top-down object identification by adding partially instantiated object linkage structures containing information about a symbol that the user would like to have anchored. The anchoring process then attempts to identify these objects by processing incoming percepts in an attempt to satisfy the reestablish condition and extend the partial structures until they are anchored. This is an interesting topic for additional future work.

3 DyKnow

The anchoring framework has been realized by extending the stream-based knowledge processing middleware framework DyKnow [10, 14]. DyKnow helps organize the many levels of information and knowledge processing in a robotic system as a coherent network of processes connected by streams. The streams contain time-stamped information and can be viewed as continually evolving time-series. In addition to providing conceptual support, DyKnow serves as a central component in the UASTech UAV architecture [6].

A knowledge processing application in DyKnow consists of a set of *knowledge processes* connected by *streams* satisfying *policies*. A policy is a declarative specification of the desired properties of a stream. Each knowledge process is an instantiation of a *source* or *computational unit* providing *stream generators* that produce streams. A source makes external information available in the form of streams while a computational unit refines and processes streams. A formal language called KPL is used to write declarative specifications of DyKnow applications (see [10, 14] for details).

DyKnow views the world as consisting of *objects* and *features* representing for example properties of objects and relations between objects. A *sort* is a collection of objects, which may represent that they are all of the same type.

Due to inherent limitations in sensing and processing, an agent cannot always expect access to the actual value of a feature over time. Instead it has to use approximations. Such approximations are represented as streams of *samples* called *fluent streams*. Each sample represents an observation or estimation of the value of a feature at a specific point in time called the *valid time*. A sample is also tagged with its *available time*, the time when it is ready to be processed by the receiving process after having been transmitted through a potentially distributed system. This allows us to formally model delays in the availability of a value and permits an application to use this information introspectively to determine whether to reconfigure the current processing network to achieve better performance. DyKnow also provides support for generating streams of *states* by synchronizing distributed individual streams. Using the stream specifications it can be determined when the best possible state at each time-point can be extracted [10]. This is essential for evaluating the temporal conditions used in anchoring. DyKnow also supports *stream reasoning*, that is, incremental reasoning over streams. For example, it incrementally evaluates logic formulas given a stream of states.

For the purpose of anchoring (and several other tasks) we begin with first order logic, a powerful technique for expressing complex relationships between objects, and extend it with operators allowing metric temporal relationships to be expressed. Specifically, $\Diamond_{[\tau_1,\tau_2]}\phi$ ("eventually ϕ") is true at time τ iff ϕ holds at *some* $\tau' \in [\tau+\tau_1, \tau+\tau_2]$, allowing us to model conditions such as "formula F must become true within 30 seconds". Also, $\Box_{[\tau_1,\tau_2]}\phi$ ("always ϕ") is true at τ iff ϕ is true at *all* $\tau' \in [\tau+\tau_1, \tau+\tau_2]$, allowing us to say that a formula F' must hold in every state between 10 and 20 seconds from now. Finally, $\phi \mathsf{U}_{[\tau_1,\tau_2]}\psi$ ("until") is true at τ iff ψ is true at some $\tau' \in [\tau+\tau_1, \tau+\tau_2]$ such that ϕ is true in all states in (τ, τ'): Essentially, ϕ must be true until ψ becomes true, which must happen in the specified interval. As exemplified later, these operators can be nested. This is similar to the well known Metric Temporal Logic [18], but has here been integrated as a part of Temporal Action Logic (TAL) [5]. To support spatio-temporal stream reasoning, the progression algorithm has also recently been extended with support for spatial reasoning in RCC-8 [19].

The semantics of metric temporal formulas is defined over infinite state sequences. To make the logic suitable for stream reasoning, formulas are incrementally evaluated using *progression* over a stream of timed states, allowing them to be processed in real time as states become available.

The result of progressing a formula through the first state in a stream is not a truth value but a new formula that holds in the remainder of the state stream if and only if the original formula holds in the complete state stream. In essence, all parts of a temporal formula that refer to a particular state are resolved against that state when it arrives and never have to be re-evaluated. If progression returns the formula true (false), then the information provided in the states through which the original formula has been progressed was sufficient to determine that the formula must be true (false), regardless of future states. For example, to verify $\Diamond_{[0,\infty]}\phi$, it is sufficient that at *some* time we receive a state where ϕ is true, and what happens after that cannot matter.

Even though the size of a progressed formula may grow exponentially in the worst case, it is always possible to use bounded intervals to limit the potential growth. A variety of formula simplification rules can also be applied to the progressed formula, further limiting growth [10].

DyKnow has been realized both using CORBA [10] and in the Robot Operating System (ROS) [15]. Figure 2 gives an overview of the stream reasoning architecture as it is realized in ROS and the steps involved in logic-based spatio-temporal stream reasoning. An important feature of this architecture is the semantic matching functionality that matches symbols to streams based on their meaning [11]. This is achieved by creating a common ontology, specifying the semantic content of streams relative to the ontology and then using semantic matching to find relevant streams. By using semantic mappings between ontologies it is also possible to do semantic matching over multiple ontologies.

The semantic matching is used to find streams containing the information necessary for the evaluation of a particular formula. When suitable streams have been found, they are connected to a stream processing engine that regularly generates temporally synchronized states, based on information received, to be used in formula evaluation.

3.1 Anchoring in DyKnow

Object linkage structures are created in DyKnow using a particular type of knowledge process called a *Link Process*. This process handles linking between two specific types of entities (A and B), either a percept type and an object type or two object types. An example link process is shown in Fig. 3 where vision percepts are linked to car objects.

The link process subscribes to a stream containing information about all objects of type A. It then uses DyKnow's support for formula progression to evaluate the associated establish, reestablish, and maintain conditions over the temporal evolution of each of these objects, thereby determining when to create and remove links (the bottom part of the figure). Concurrently, one DyKnow computational unit is used for each current link $A \to B$ to calculate the state of the object of type B from the state of the corresponding object of type A, possibly together with additional information from other sources such as a geographic information system. The output of the link process contains information about all objects currently hypothesized as being of type B. More details about Link Processes can be found in Heintz et al. [13].

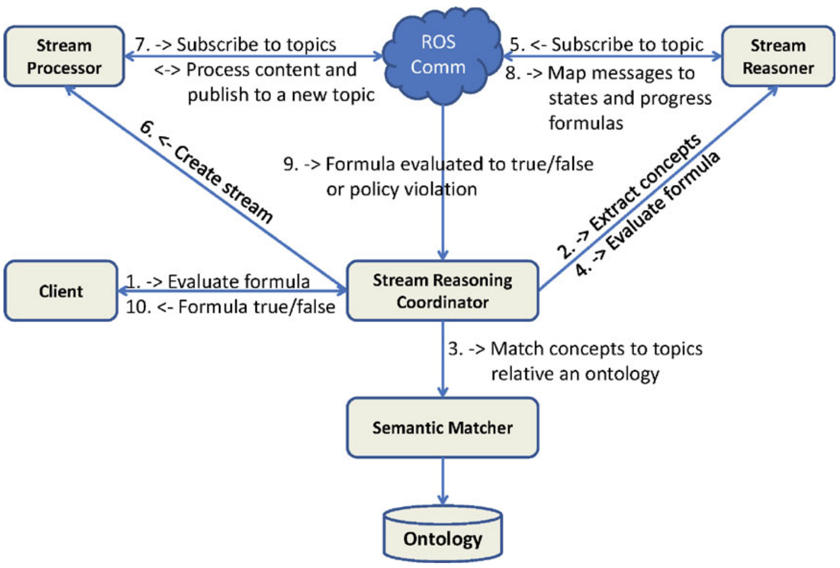

Stream Processor
7. -> Subscribe to topics
<-> Process content and publish to a new topic
ROS Comm
5. <- Subscribe to topic
8. -> Map messages to states and progress formulas
Stream Reasoner
6. <- Create stream
9. -> Formula evaluated to true/false or policy violation
2. -> Extract concepts
4. -> Evaluate formula
Client
1. -> Evaluate formula
10. <- Formula true/false
Stream Reasoning Coordinator
3. -> Match concepts to topics relative an ontology
Semantic Matcher
Ontology

Fig. 2 The DyKnow stream reasoning architecture

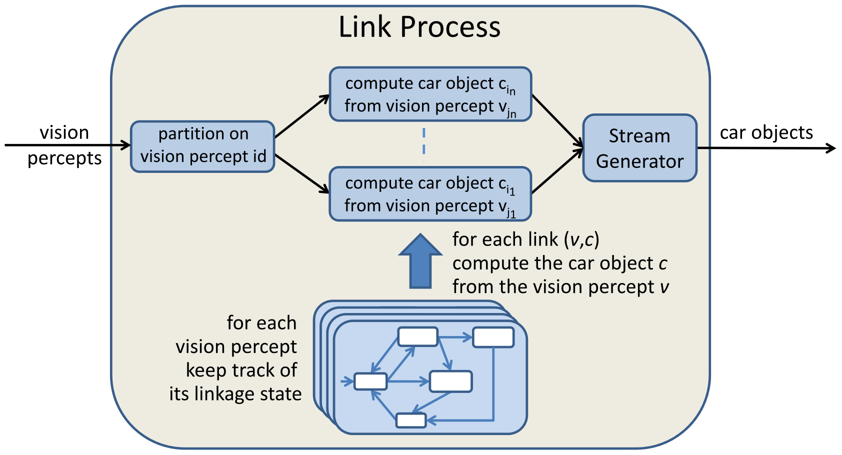

Link Process
vision percepts
partition on vision percept id
compute car object c_{in} from vision percept v_{jn}
compute car object c_{i1} from vision percept v_{j1}
Stream Generator
car objects
for each link (v,c) compute the car object c from the vision percept v
for each vision percept keep track of its linkage state

Fig. 3 A link process creating car objects from vision percepts

One major issue is how to handle the fact that after linking two object structures their symbols should be considered equivalent. If wo3 is linked to oo1, subscribing to speed(wo3) should be equivalent to subscribing to speed(oo1). To support this, which was not done before, we leverage the semantic matching functionality of DyKnow. We therefore propose using an ontology to represent both the specification of object linkage structures and what percepts and objects are currently linked and thereby equivalent.

Ontologies provide support for creating and reasoning with machine readable domain models [16]. In the ontology each percept and object type is represented as a *class*. The ontology has two class hierarchies, one for percept types and one for object types. To facilitate improving the classification of an object, each object class has an establish and a reestablish condition relative to its ancestor represented as properties. Multiple inheritance is currently not allowed. To facilitate linking between different class hierarchies, such as percepts to object, a new class hierarchy is introduced with the top class Link. Each class in the Link hierarchy represents one link type, such as VisionPerceptToWorldObjectLink, and has properties representing the establish and reestablish conditions. To continually check that an entity is an instance of its class every class has a property representing its maintenance condition. The reason for representing each link type as a class in the ontology is to support reasoning about link types at the terminological (T-Box) level. Only reasoning about equivalences among object and percept instances is done at the assertion (A-Box) level.

From a realization perspective, this replaces the previous KPL link specification. An important change to the Link Process is that each time two entities are linked a SameIndividual assertion is added to the ontology and each time a link is removed the corresponding assertion is removed. These changes affect future subscriptions. We are working on allowing subscriptions to change dynamically.

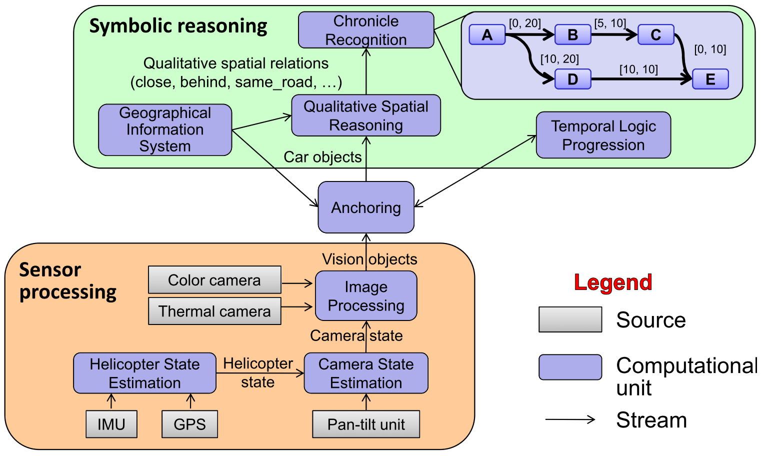


Fig. 4 Incremental processing for a traffic monitoring task

4 A UAV Traffic Monitoring Case Study

The anchoring framework presented above can be applied in a wide variety of robotic applications where there is a need to reason symbolically over extended periods of time about objects perceived by sensors.

To illustrate the approach, we again consider the traffic monitoring scenario discussed in the introduction. Here traffic violations and other events to be detected should be represented formally and declaratively, which is a specific example of a general classification task which is common in robotic applications. This can be done using *chronicle recognition* [8], where each chronicle defines a parameterized class of complex events as a simple temporal network [4] whose nodes correspond to occurrences of high-level qualitative events and edges correspond to metric temporal constraints between event occurrences. For example, events representing changes in qualitative spatial relations such as $\mathsf{beside}(car_1, car_2)$, $\mathsf{close}(car_1, car_2)$, and $\mathsf{on}(car_1, road_7)$ might be used to detect a reckless overtake [12].

Creating these high-level representations from low-level sensor data, such as video streams from the color and thermal cameras on-board our UAVs, involves extensive information processing within each sensor platform. Figure 4 provides a partial overview of the incremental processing required for a traffic monitoring task implemented as a set of distinct DyKnow knowledge processes. Anchoring is a central process making symbolic reasoning about the external world possible by creating symbols referring to objects in the world based on processing of sensor data. The anchoring process is actually a set of concurrent link processes as described in the previous section.

At the lowest level, a *helicopter state estimation component* uses data from an *inertial measurement unit* (IMU) and a *global positioning system* (GPS) to determine the current position and attitude of the UAV. A *camera state estimator* uses this information, and the current state of the *pan-tilt unit* on which the cameras are mounted, to generate information about the current camera state. *Image processing* uses this to determine where the camera is currently pointing. Video streams from the *color* and *thermal cameras* can then be analyzed in order to generate *vision percepts* representing hypotheses about moving and stationary physical entities, including their approximate positions and velocities.

Symbolic formalisms such as chronicle recognition [8] require a consistent assignment of symbols, or identities, to the physical objects being reasoned about and the sensor data received about those objects. Image analysis provides a partial solution, with vision percepts having symbolic identities that persist over short intervals of time. However, changing visual conditions or objects temporarily being out of view lead to problems that image analysis often cannot (and should not) handle. This is the task of the anchoring system, which also assists in object classification and in the extraction of higher level attributes of an object. A *geographic information system* is used to determine whether an object is currently on a road. Such attributes can in turn be used to derive relations *between* objects, including *qualitative spatial relations* such as $\mathsf{beside}(car_1, car_2)$ and $\mathsf{close}(car_1, car_2)$. Concrete events corresponding to changes in such attributes and predicates finally provide sufficient information for the *chronicle recognition system* to determine when higher-level events such as reckless overtakes occur.

In the case study, anchoring links vision percepts from an object tracker to world objects, which are then linked to on-road objects. Link conditions below are intended to demonstrate key concepts and could be elaborated to take more complex conditions into account. The temporal unit in the formulas is milliseconds, which is the temporal granularity used in the real system.

We used the formula $\Diamond_{[0,1000]} \mathsf{xydist}(from, to) < \mathsf{thresh}$ as a reestablish condition from vision percepts to existing world objects: The distance to the (predicted or simulated)

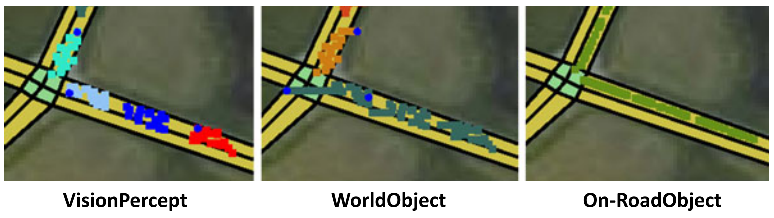


Fig. 5 A demonstration of the result of hierarchical anchoring

world object in the x/y plane must be within a given threshold within one second. Since we believe all vision percepts do correspond to world objects, the corresponding establish condition is simply $\Diamond_{[1000,\infty]}$ `true`, which unconditionally becomes true to generate a new world object if after one second no reestablish condition has been satisfied. Finally, the maintain condition is `true`: As long as the vision percept persists, we do not doubt its identity as a world object.

A link from a world object to an existing on-road object is reestablished if $\Diamond_{[0,1000]}\,\mathsf{xydist}(\mathit{from},\mathit{to}) < \mathsf{thresh}$. Not all world objects are on-road objects. A new on-road object is therefore only created from a world object if $\Diamond_{[1000,\infty]}\,\Box_{[0,1000]}\,\mathsf{on\_road}(\mathit{from})$, that is, if within one second, the object is detected on a road for at least one second. Finally, the maintain condition is $\Box\,\Diamond_{[0,30000]}\,\Box_{[0,10000]}$ $\mathsf{on\_road}(\mathit{from})$. In other words, it must always be the case that within 30 seconds, the object is detected as being on a road for at least 10 seconds. These temporal intervals are adjusted to suit the specific noise profile of our simulated sensors with a minimum of false positives or negatives.

Straight-forward simulations were used for temporarily unanchored world and on-road objects. World objects are assumed to continue traveling in the same direction, and with the same speed, as when they were last anchored. On-road objects are also assumed to retain their speed, but travel along the road network, splitting into multiple objects at crossings. This greatly improves the chance of reacquiring an anchor when the roads are curved or contain crossings.

Figure 5 shows a representative example from running a car tracking simulation with noisy sensors with four tracks, from the same car, each about 3 seconds long and then roughly 2 seconds without any tracking. The speed of the car is 15 meters/second and the sample period is 100 milliseconds. Uniform noise in the range -3 to 3 meters is added to the x and y coordinates independently, which is roughly the quality of our current image-based tracker. Each dot represents an estimated position of an object. Each color represents a separate object identity. The vision percept positions come directly from image processing and are missing where image processing fails to detect a vehicle. The computational unit that continually updates an on-road object from its associated world object knows that an on-road object tends to travel in the center of a specific lane, and adjusts positions accordingly (right most image). When vision percepts are missing, position estimates for a world object are provided by simulation, assuming a continuous straight line motion. Position estimates for an on-road object, on the other hand, are simulated based on the assumption that the object continues along the road with constant speed. When a simulated on-road object enters a crossing, one hypothesis for each potential next road segment is generated to qualitatively represent the uncertainty in the position. The difference is clearly seen in the crossing, where the estimated world object continues straight out into the grass on the other side of the crossing while the on-road object follows the road and can be matched to the new vision percept after the crossing.

For more details about the traffic monitoring application see Heintz et al. [12] and for more details about how the anchoring process is realized in DyKnow see Heintz [10] and Heintz et al. [13].

5 Related Work

In recent research, anchoring has been considered as a separate problem, which has generated a series of interesting approaches [1–3, 7, 9, 20, 22, 23].

One well-known framework was proposed by Coradeschi and Saffiotti [3]. Their approach converts the quantitative attributes of a percept to a set of symbolic predicates using a *predicate grounding relation*. An object described by a set of predicates can be found by finding a percept whose attributes match the predicates of the object according to the predicate grounding relation. Objects are tracked using a single step prediction function computing the next state of the object which can then be compared with the next percept. As long as the predictions and the observations match, the object is tracked. Reacquiring an anchor is similar to acquiring it, but the information collected about the object while it was tracked can be used to improve the re-identification.

In more recent work this approach has been extended to allow multiple percepts to be anchored to a single symbol [21], to support the anchoring of not only unary predicates but also binary relations [21], and to actively control a robot with the goal of anchoring particular symbols [17].

In our framework, predicate grounding relations can be encoded in link specifications. Thanks to the use of a metric temporal logic, we can also use anchoring conditions that range over several observations. This can be used to model arbitrary tradeoffs between false positives and false negatives in the case where a percept may *temporarily* fail to satisfy a condition, which is not easily doable in their framework. Another benefit of our approach is the possibility to do anchoring in several smaller steps. Instead of describing how to directly identify a particular percept as a specific car we can connect a percept to a world object, which can be connected to an on road object, which can finally be connected to a car object. This allows the predicate grounding relation to be contextual, based on the types of objects being linked.

Another related framework is the GLAIR grounded layered architecture with integrated reasoning [22]. GLAIR is a three-layered architecture consisting of a knowledge level (KL), a perceptuo-motor level (PML), and a sensori-actuator level (SAL). Anchoring is done by aligning KL terms representing mental entities with PML descriptions representing perceived objects. A PML description is an n-tuple where each component is a value from some perceptual feature domain. This alignment is made by hand. If the SAL finds an object with a particular PML description and there is only one KL term aligned with it, then the term is anchored to the PML description and indirectly to the external physical object. With the exception that GLAIR has two clearly distinct levels and an anchor does not belong to either of them, this approach is similar to the one by Coradeschi and Saffiotti.

Another approach is used by Steels and Baillie [23] to achieve shared grounding among agents. In their experiment, two robots watch the same scene using cameras and try to agree what is happening through the use of natural language. Several image processing and pattern matching techniques are used to interpret the scene and to ground the natural language utterances. Image processing is used to classify objects found in video sequences and to collect qualitative information about the objects such as their shape and color in the form of predicate logical statements. Then pattern matching techniques are used to detect first changes in the properties of objects and then events as sequences of such changes. The main difference to our approach is the way the identity of individual objects are determined. Instead of using predefined histograms, we describe the conditions for when two objects should be assumed to be identical using a metric temporal logic, an approach we believe is more flexible and general.

A fourth related approach proposes a method for anchoring symbols denoting composite objects through anchoring the symbols of their corresponding component objects [7]. This extends the Coradeschi and Saffiotti framework with the concept of a composite anchor, which is an anchor without a direct perceptual counterpart. The composite anchor computes its own perceptual signature from the perceptual signatures of its component objects. The benefit is that each sensor can anchor its sensor data to symbols which can be used to build composite objects fusing information from several sensors. The same functionality is provided by our approach, since objects do not have to have direct perceptual counterparts but can be computed from other objects which may or may not acquire their input directly from sensors.

Compared to existing approaches our approach supports hierarchical anchoring where an object is incrementally anchored to more and more abstract objects. For example, in the traffic monitoring scenario we start by linking blobs found by an image processing system, representing potential cars, to world objects. These world objects can then be linked to on-road objects, which can finally be linked to car objects. Each step in this chain adds more abstract attributes to the object and allows for more specific assumptions to be made about how such an object behaves. These assumptions can be used when trying to reacquire an anchor by predicting where the object is.

Another significant difference is that our approach explicitly models time using a metric temporal logic, allowing the use of complex temporal conditions to decide when to anchor symbols to objects. This is especially important when anchoring dynamic objects such as people and vehicles.

6 Conclusions

We have presented a general stream-based hierarchical anchoring framework and an implementation as an extension of the DyKnow knowledge processing middleware. The framework dynamically creates object linkage structures representing the current hypothesis about the identity and classification of an object. As more information becomes available the structures are updated either by narrowing down the classification or by removing violated hypotheses. The result is that each object linkage structure maintains the best possible hypothesis about the identity and classification of a single physical object. The symbol associated with an object in the object linkage structure can then be used to further reason about the object.

The use of an expressive metric temporal logic and the support for hierarchical anchoring gives our approach qualitative advantages over existing systems. First, the hierarchical approach is a strict extension since it is still possible to do direct anchoring. Second, hierarchies support a divide and conquer approach to describing how to anchor an object which also supports reuse. Third, it is possible to have multiple coherent representations of the same object allowing different functionalities to interact with the object at the appropriate level of abstraction. Finally, the use of an expressive metric temporal logic allows complex temporal relations to be used in the anchoring process. The approach has been applied to a traffic monitoring application where a UAV detects traffic violations.

References

1. Bonarini A, Matteucci M, Restelli M (2001) Anchoring: do we need new solutions to an old problem or do we have old solutions for a new problem? In: Anchoring symbols to sensor data in single and multiple robot systems: AAAI Fall symposium
2. Chella A, Frixione M, Gaglio S (2003) Anchoring symbols to conceptual spaces: the case of dynamic scenarios. Robot Auton Syst 43(2–3):175–188
3. Coradeschi S, Saffiotti A (2003) An introduction to the anchoring problem. Robot Auton Syst 43(2–3):85–96
4. Dechter R, Meiri I, Pearl J (1991) Temporal constraint networks. Artif Intell 49:61–95
5. Doherty P, Kvarnström J (2008) Temporal action logics. In: Handbook of knowledge representation. Elsevier, Amsterdam
6. Doherty P, Haslum P, Heintz F, Merz T, Nyblom P, Persson T, Wingman B (2004) A distributed architecture for autonomous unmanned aerial vehicle experimentation. In: Proc 7th international symposium on distributed autonomous robotic systems (DARS), pp 221–230
7. Fritsch J, Kleinehagenbrock M, Lang S, Plötz T, Fink GA, Sagerer G (2003) Multi-modal anchoring for human-robot interaction. Robot Auton Syst 43(2–3):133–147
8. Ghallab M (1996) On chronicles: representation, on-line recognition and learning. In: Proc KR
9. Gunderson JP, Gunderson LF (2006) Reification: what is it, and why should I care? In: Proc PerMIS
10. Heintz F (2009) DyKnow: a stream-based knowledge processing middleware framework. PhD thesis, Linköpings Universitet
11. Heintz F, Dragisic Z (2012) Semantic information integration for stream reasoning. In: Proc fusion
12. Heintz F, Rudol P, Doherty P (2007) From images to traffic behavior—a UAV tracking and monitoring application. In: Proc fusion
13. Heintz F, Kvarnström J, Doherty P (2009) A stream-based hierarchical anchoring framework. In: Proc of IROS
14. Heintz F, Kvarnström J, Doherty P (2010) Bridging the sense-reasoning gap: DyKnow—stream-based middleware for knowledge processing. Adv Eng Inform 24(1):14–26
15. Hongslo A (2012) Stream processing in the robot operating system framework. Master's thesis, Linköpings Universitet
16. Horrocks I (2008) Ontologies and the Semantic Web. Commun ACM 51(12):58. doi:10.1145/1409360.1409377
17. Karlsson L, Bouguerra A, Broxvall M, Coradeschi S, Saffiotti A (2008) To secure an anchor—a recovery planning approach to ambiguity in perceptual anchoring. AI Commun 21(1):1–14
18. Koymans R (1990) Specifying real-time properties with metric temporal logic. Real-Time Syst 2(4):255–299
19. Lazarovski D (2012) Extending the stream reasoning in DyKnow with spatial reasoning in RCC-8. Master's thesis. Linköpings Universitet
20. Lemaignan S, Ros R, Sisbot EA, Alami R, Beetz M (2011) Grounding the interaction: anchoring situated discourse in everyday human-robot interaction. Int J Soc Robot 1–19
21. Loutfi A, Coradeschi S, Daoutis M, Melchert J (2008) Using knowledge representation for perceptual anchoring in a robotic system. Int J Artif Intell Tools 17(5):925–944
22. Shapiro SC, Ismail HO (2003) Anchoring in a grounded layered architecture with integrated reasoning. Robot Auton Syst 43(2–3):97–108
23. Steels L (2003) Shared grounding of event descriptions by autonomous robots. Robot Auton Syst 43(2–3):163–173

Fredrik Heintz is a researcher at the Department of Computer and Information Science (IDA), Linköping University, Sweden, where he currently leads the Embedded Reasoning Systems Group. He completed his Ph.D. in 2009. He is currently the president of SAIS, the Swedish Artificial Intelligence Society. His main research areas are knowledge representation, information fusion, intelligent autonomous systems, and multi-agent systems.

Jonas Kvarnström is a researcher at the Department of Computer and Information Science (IDA), Linköping University, Sweden, where he currently leads the Automated Planning Group. He completed his Ph.D. in 2005. His main research areas are knowledge representation, intelligent autonomous systems, and automated planning.

Patrick Doherty is a Professor of Computer Science at the Department of Computer and Information Science (IDA), Linköping University, Sweden. He heads the Artificial Intelligence and Integrated Computer Systems Division at IDA. He is currently President of ECCAI, the European Coordinating Committee for Artificial Intelligence and is an ECCAI fellow. His main areas of interest are knowledge representation, automated planning, intelligent autonomous systems and multi-agent systems.