A Stream-Based Hierarchical Anchoring Framework

Fredrik Heintz, Jonas Kvarnström and Patrick Doherty
Department of Computer and Information Science, Linköpings universitet,
SE-581 83 Linköping, Sweden
{frehe, jonkv, patdo}@ida.liu.se

***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 labeling physical objects and the sensor data being collected about them, a process called *anchoring*.**

In this paper we present a stream-based hierarchical anchoring framework extending the DyKnow knowledge processing middleware. 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 has been applied to a traffic monitoring application where an unmanned aerial vehicle collects information about a small urban area in order to detect traffic violations.

I. 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 is how to create and maintain a consistent correlation between symbolic representations of these objects and sensor data that is being continually collected about them. This process is called *anchoring* [3].

As a motivating example, consider an unmanned aerial vehicle (UAV) using color and thermal cameras to monitor a small urban area for potential or ongoing criminal activities. In the case of traffic violations, the UAV must be able to use the available video feeds to recognize and track cars and other vehicles. Initial recognition involves taking a picture of an area and correctly identifying any vehicles in that area. To track the vehicles, the UAV must continually analyze the video feed and keep its information up to date. This requires the ability to determine if a car found in a frame is a previously unseen car or if it has been seen before.

Tracking an object, such as a car, through a series of images is a classical problem. There are many effective solutions for the case where the object is easily distinguishable and 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 may be present in the world. In this case, pure image-based tracking does not provide a complete solution, since it usually only considers the information available in the image itself. A more suitable approach would be to also include 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 would include the layout of the road network and the typical size, speed, and driving behavior of cars. It has been argued that anchoring is an extension to classical tracking approaches which handles missing data in a principled manner [7].

In this paper we propose a general stream-based hierarchical anchoring framework as an extension of our existing knowledge processing middleware framework DyKnow [10–12]. The proposed solution uses *object linkage structures* to incrementally classify and track objects found by image processing, radar sensors, or similar techniques, and to anchor these objects to symbolic identifiers.

II. DyKnow

DyKnow is a stream-based knowledge processing middleware service that 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 is fully implemented and serves as a central component in the UASTech UAV architecture [5].

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 generate 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, 11] for details). The DyKnow service, which implements the DyKnow framework, takes a set of KPL declarations and sets up the required processing and communication infrastructure. Due to the use of CORBA [18] for communication, knowledge processes are location-agnostic, providing

This work is partially supported by grants from the Swedish Foundation for Strategic Research (SSF) Strategic Research Center MOVIII, the Swedish Research Council (VR) Linnaeus Center CADICS, and the Center for Industrial Information Technology CENIIT (06.09).

support for distributed architectures running on multiple networked computers.

III. A Traffic Monitoring Scenario

Examples in this paper will be taken from a specific scenario involving UAV-based traffic monitoring. However, we emphasize that this approach to anchoring is general and 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.

Suppose a human operator is trying 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 also 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.

Traffic violations, or other events to be detected, should be represented formally and declaratively. 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. Creating these high-level representations from low-level sensor data, such as video streams from color and thermal cameras, involves extensive information and knowledge processing within each sensor platform.

Fig. 1 provides an overview of how part of the incremental processing required for the traffic surveillance task could be organized as a set of distinct DyKnow knowledge processes. 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 estimation component* uses this information, together with the current state of the *pan-tilt unit* on which the cameras are mounted, to generate information about the current camera state. The *image processing component* uses the camera state 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 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 may provide 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 cannot (and should not) handle. This is the task of the anchoring system to be described in the next section, which uses *progression* of formulas in a metric temporal logic to evaluate potential hypotheses about the observed objects. The anchoring system also assists in object classification and in the extraction of higher level attributes of an object. For example, a *geographic information system* can be used to determine whether an object is currently on a road or in a crossing. 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.

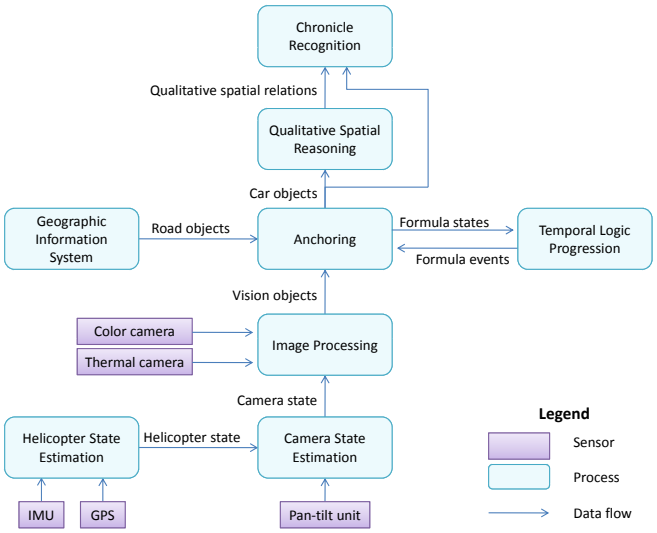


Fig. 1. An overview of how the incremental processing required for the traffic surveillance task could be organized.

Details about the image processing and the recognition of traffic violations can be found in [13].

IV. Anchoring using Object Linkage Structures

The objective of the anchoring process is to connect symbols to sensor data originating in the physical world. This requires processing on many abstraction levels, including specialized techniques adapted to a particular type of sensor. Such techniques are often well suited to determining which parts of the raw sensor input pertain to a single physical object, and may even 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 but whose identity persist. A *vision percept*, for example, could include color and size information for a moving or stationary blob. We may also have *radar percepts* or *laser range finder percepts*, which might refer to 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 (for some exceptions see the related work section), we define an incremental process of object classification and (re-)identification. This process builds on a hierarchy of percept / object types.

An example hierarchy for the traffic monitoring scenario can be seen in Fig. 2. A *world object* represents a physical object in the world. 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 could either be *on-road objects* moving along roads or *off-road objects* not following roads. 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*, or a *truck*. 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. An example is shown in Fig. 3, representing the hypothesis that vision percept vp8, world object wo5, on-road object oo3, and car object co2 all correspond to the same physical object.

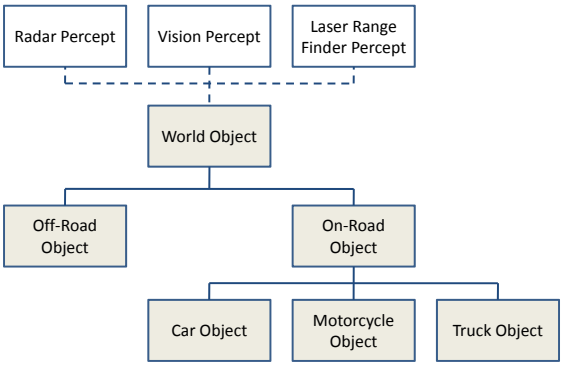


Fig. 2. The example percept (white) / object (gray) hierarchy used in the traffic monitoring scenario.

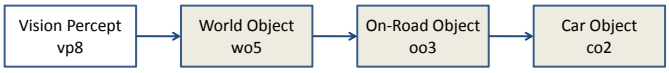


Fig. 3. An example object linkage structure.

A. Incremental Generation of Object Linkage Structures

Whenever a new object of a given type is generated, it must be determined whether it also belongs to a particular subtype in the hierarchy. For example, a new vision percept originating in image processing may be classified as corresponding to a world object. In this case, it must also be linked to a world object structure, thereby generating an object linkage structure. 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. Thus, objects and their symbols are not immediately removed when their associated percepts disappear, and any new object at one level might correspond to either a new object or an *existing* object at the next level. To reduce the computational cost, objects which are not likely to be found again are removed. Currently we discard objects which have not been observed or anchored for a certain application-dependent time.

Three conditions are used to determine when to add and remove links between objects belonging to types A and B. These conditions are written in an expressive temporal logic, similar to the well known Metric Temporal Logic [15], and incrementally evaluated by DyKnow using progression over a timed state sequence.[1] Informally, $\Diamond_{[\tau_1,\tau_2]}\,\phi$ ("eventually") holds at τ iff ϕ holds at some $\tau' \in [\tau+\tau_1, \tau+\tau_2]$, while $\Box_{[\tau_1,\tau_2]}\,\phi$ ("always") holds at τ iff ϕ holds at all $\tau' \in [\tau+\tau_1, \tau+\tau_2]$. Finally, $\phi\,\mathsf{U}_{[\tau_1,\tau_2]}\,\psi$ ("until") holds at τ iff ψ holds at some $\tau' \in [\tau+\tau_1, \tau+\tau_2]$ such that ϕ holds in all states in (τ, τ'). See [6] for formal details.

The unary *establish condition* expresses when an object of type A, which may be a percept or a higher level object, should be linked to a *new* object of type B. When a new ob-

[1]Progression incrementally evaluates formulas in a state sequence. The result of progressing a formula through the first state in a sequence is a new formula that holds in the remainder of the state sequence iff the original formula holds in the complete state sequence. If progression returns true (false), the entire formula must be true (false), regardless of future states.

ject of type A is created, the anchoring system immediately begins evaluating this condition. If and when the condition becomes true, a link to a new object of type B is created. Concrete examples will be given below. This corresponds to the Find functionality suggested by Coradeschi and Saffiotti [3], which takes a symbolic description of an object and tries to anchor it in sensor data.

The binary *reestablish condition* expresses the condition for an object of type A to be linked to a *known* object of type 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. When a new object of type A is created, the anchoring system immediately begins to evaluate the re-establish condition for every known object of type B that is not 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 [3].

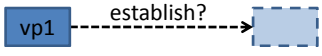
While two objects are linked, the attributes of the more specific object are computed using a DyKnow computational unit from the attributes of the less specific object, possibly together with information from other sources.

Finally, since observations are uncertain and classification is imperfect, any link created between two objects 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.

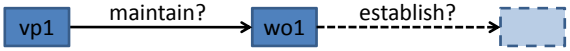
If a maintain condition is violated, the corresponding link is removed. However, all objects involved remain, enabling re-classification and re-identification at a later time. This corresponds to the Track functionality [3].

The state of an object having no incoming links will be predicted based on a general model of how objects of this type normally behave. In future work, we may extend this ability by estimating a special model for this particular individual using system identification techniques [16] on the data collected while tracking it.

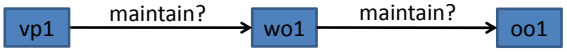
Example IV.1 Assume that the image processing system is currently tracking a potential car represented by the vision percept vp1 and that no other objects have been created. Since there is a vision percept but no known world objects it is enough to evaluate the establish condition on vp1.



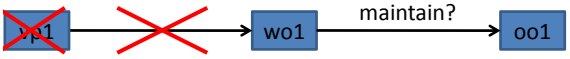
If the establish condition is eventually satisfied, a new world object wo1 is created which is associated with vp1. As long as wo1 is associated with vp1 its state will be computed from the state of the vision percept vp1. It is also possible to estimate a model of the behavior of wo1 using the collected information. This model can later be used to predict the behavior of wo1 if it is no longer tracked. The maintain condition is monitored to continually verify the hypothesis that wo1 is a world object.



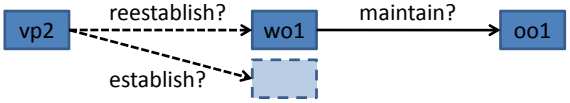
Further, assume that after a while wo1 has been hypothesized as being an on-road object represented by oo1 and an object linkage structure has been created containing all three objects. For example, one could assume that an object is an on-road object after it has been observed on a road for at least 30 seconds.



Assume the image processing system loses track of the potential car after a while. Then vp1 is removed together with the link to wo1. Even though the link is removed the world object wo1 remains as well as the on-road object oo1 and its link to wo1. While wo1 is not linked to any vision percept its state will be 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 makes it much easier to reestablish a link to it.



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



Assume that after a while 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. To verify the new hypothesis a maintain condition between wo1 and vp2 is monitored.

□

Top-down Identification. As shown above, object linkage structures can be created bottom-up by processing streams of incoming percepts, creating a new symbol for each hypothesized object found in the stream. A similar approach can be used for top-down object identification by adding partially

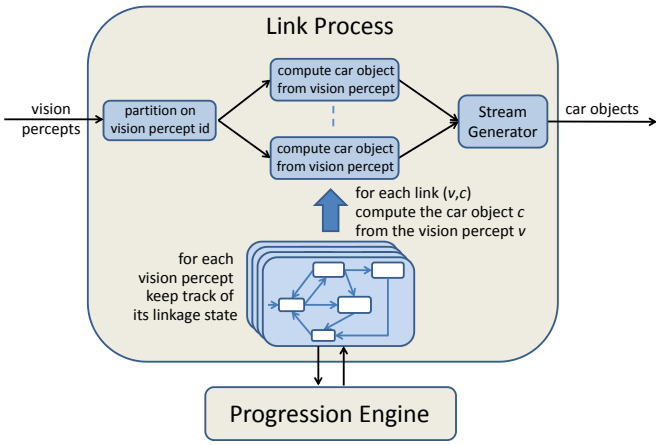


Fig. 4. A link process creating car objects from vision percepts.

instantiated object linkage structures containing information about a symbol that the user would like to have anchored. The anchoring process will then try 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. However, more work is needed and this will be an interesting topic for future work.

B. Link Specifications and Link Processes

Though the ideas used in this anchoring framework can be used in other settings, the particular instance described here is adapted to the use of knowledge processes and streams in the DyKnow framework.

Thus, every type of percept originates in a particular knowledge process, such as an image processing knowledge process for vision percepts. Subscribing to the output of this process generates a stream of vision percepts, where new percepts are sent as soon as they are available.

Similarly, the task of classifying objects of type A as being objects of subtype B is modeled and implemented as a *link process*. This process subscribes to a stream containing information about all objects of type A. It then uses DyKnow's support for formula progression [6, 11] 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. Concurrently, the process uses a DyKnow computational unit to calculate the state of each 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. It is possible to subscribe to the output of the process to find information about all objects currently hypothesized as being of type B. An example link process is shown in Fig. 4 where vision percepts are linked to car objects. Which car object a vision percept is linked to depends on the three link conditions.

The DyKnow knowledge processing language KPL has been extended to allow the declaration of link processes. A *link specification* $\langle a, b, cu, e, r, m\rangle$ describes a knowledge process that subscribes to a process (strictly speaking, a stream generator) labeled a and links the objects received from a according to the establish condition e, the reestablish condition r, and the maintain condition m. These formulas may use the special variables *from* and *to*, which refer to the object of type A being linked from and the object of type B being linked to, respectively. Instances of the computational unit cu are used to compute the new object states. All the generated object states will be made available from a stream generator labeled b.

Conceptually, for each object p found in the input stream a of a link specification $\langle a, b, cu, e, r, m\rangle$, a new process within the link process is created and executed throughout the lifetime of p. This process does the following:

```
state = unlinked
add establish(p) to progressor
for each object o of type B { add reestablish(p,o) to progressor }
while true {
  wait for event e
  if (state == unlinked) {
    if (e == addition of unlinked object o of type B) {
      // Another potential candidate
      add reestablish(p,o) to progressor
    } else if (e == removal of existing object o of type B) {
      // Candidate disappeared
      remove reestablish(p,o) from progressor
    } else if (e == linkage of o to p' != p) {
      // Candidate linked to something else
      remove reestablish(p,o) from progressor
    } else if (e == signal from progressor: establish(p) became true) {
      // We believe p is a new object
      create new object o of type B
      link p to o using cu to calculate states
      add maintain(p,o) to progressor
      state = linked
    } else if (e == signal from progressor: reestablish(p,o) became true) {
      // We believe p corresponds to an existing object o
      link p to o using cu to calculate states
      add maintain(p,o) to progressor
      state = linked
    }
  } else { // state == linked
    if (e == signal from progressor: maintain(p,o) became false) {
      // The hypothesis that p corresponds to o is invalid
      remove link from p to o
      state = unlinked
      // Must evaluate other hypotheses
      add establish(p) to progressor
      for each object o { add reestablish(p,o) to progressor }
    }
  }
}
```

V. Implementation

The proposed approach to anchoring has been tested in a UAV traffic monitoring application, where vision percepts from a simulated object tracker are linked to world objects, which in turn are linked to on-road objects. Link conditions are intended to demonstrate key concepts and could be elaborated to take more complex conditions into account. The temporal unit is 1 millisecond.

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) 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

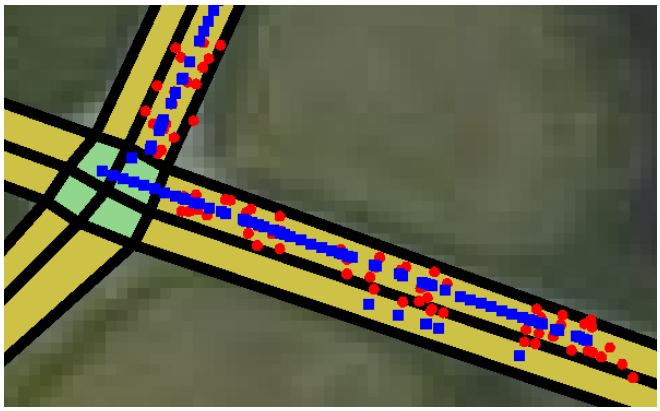


Fig. 5. One run of the car tracking simulation.

condition is simply $\Diamond_{[1000,\infty]}\,\texttt{true}$, which unconditionally triggers 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, so a new on-road object is only created from a world object if $\Diamond_{[1000,\infty]}\,\Box_{[0,1000]}\,\mathsf{on\_road}(\mathit{from})$, that is, if after one second, the object is detected on a road for at least one second. Finally, the maintain condition is $\Box\,\Diamond_{[0,10000]}\,\Box_{[0,1000]}\,\mathsf{on\_road}(\mathit{from})$. In other words, it must always be the case that within 10 seconds, the object is detected as being on a road for at least 1 second. 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 not straight or contain crossings.

Fig. 5 shows the results from running a car tracking simulation with noisy sensors with four tracks, each about 3 seconds long and then roughly 2 seconds without any tracking. Red circles indicate position estimates taken directly from vision percepts and are missing where image processing fails to detect a vehicle. The computational unit that continually updates an on-road object from its associated vision percept knows that an on-road object tends to travel in the center of a specific lane, and adjusts positions accordingly (blue squares). Additionally, when vision percepts are missing, position estimates for an on-road object are provided by a simulation of its most likely path. There are three examples of this, twice before the crossing and once inside the crossing.

The example scenario provides evidence that the framework can be used to anchor symbols in the traffic domain. A more thorough evaluation of the scalability and the range of applicability of the approach is part of our ongoing work.

VI. 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, 19, 20].

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 [17], to support the anchoring of not only unary predicates but also binary relations [17], and to actively control a robot with the goal of anchoring particular symbols [14].

In our framework, predicate grounding relations can be encoded in link specifications. Due to the use of a metric temporal logic, we can also use anchoring conditions that range over several observations. This can be use to model arbitrary tradeoffs between false positives and false negatives in the case where a percept may *temporarily* fail to satisfy a condition. Another extension 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 also allows the predicate grounding relation to be contextual, based on the types of objects being linked.

A similar approach is used by the GLAIR grounded layered architecture with integrated reasoning [19]. 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.

An approach similar to ours is used by Steels and Baillie [20] 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. To interpret the scene and to ground the natural language utterances they use several image processing and pattern matching techniques. 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 they use pattern matching techniques 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. They use predefined histograms while 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 is [7], which proposes a method for anchoring symbols denoting composite objects through anchoring the symbols of their corresponding component objects. This extends the framework presented by [3] 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 can be provided by DyKnow, 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.

This particular functionality is important to emphasize since in complex hybrid robotic architectures, different components and functionalities in the architecture require access to representations of dynamic objects in the external environment at different levels of abstraction and with different guaranteed upper bounds on latencies in data. By modeling dynamic objects as structured objects with different types of features, any functionality in the architecture can access an object at the proper level of abstraction and acquire data from the object in a timely manner.

VII. Conclusions

We have presented a stream-based hierarchical anchoring framework which is 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 approach has been applied to a traffic monitoring application where a UAV collects information about a small urban area in order to detect traffic violations.

Compared to existing approaches we support 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 improvement is that we use a metric temporal logic to represent the conditions for when to anchor symbols to objects.

References

[1] A. Bonarini, M. Matteucci, and M. Restelli. 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: Papers from the AAAI Fall Symposium*, 2001.

[2] A. Chella, M. Frixione, and S. Gaglio. Anchoring symbols to conceptual spaces: the case of dynamic scenarios. *Robotics and Autonomous Systems*, 43(2–3):175–188, 2003.

[3] S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2–3):85–96, 2003.

[4] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[5] P. Doherty, P. Haslum, F. Heintz, T. Merz, P. Nyblom, T. Persson, and B. Wingman. A distributed architecture for autonomous unmanned aerial vehicle experimentation. In *Proc. DARS*, pages 221–230, 2004.

[6] Patrick Doherty, Jonas Kvarnström, and Fredrik Heintz. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 2009.

[7] J. Fritsch, M. Kleinehagenbrock, S. Lang, T. Plötz, G. A. Fink, and G. Sagerer. Multi-modal anchoring for human-robot interaction. *Robotics and Autonomous Systems*, 43(2–3):133–147, 2003.

[8] Malik Ghallab. On chronicles: Representation, on-line recognition and learning. In *Proc. KR*, pages 597–607, 1996.

[9] J. P. Gunderson and L. F. Gunderson. Reification: What is it, and why should I care? In *Proc. PerMIS*, pages 39–46, 2006.

[10] F. Heintz, J. Kvarnström, and P. Doherty. Bridging the sense-reasoning gap: DyKnow – stream-based middleware for knowledge processing. *J. of Advanced Engineering Informatics*, accepted for publication.

[11] Fredrik Heintz. *DyKnow: A Stream-Based Knowledge Processing Middleware Framework*. PhD thesis, Linköpings universitet, 2009. Linköping Studies in Science and Technology, Dissertation No 1240.

[12] Fredrik Heintz, Jonas Kvarnström, and Patrick Doherty. Knowledge processing middleware. In *Proc. SIMPAR*, pages 147–158, 2008.

[13] Fredrik Heintz, Piotr Rudol, and Patrick Doherty. From images to traffic behavior – a UAV tracking and monitoring application. In *Proc. Fusion*, 2007.

[14] L. Karlsson, A. Bouguerra, M. Broxvall, S. Coradeschi, and A. Saffiotti. To secure an anchor - a recovery planning approach to ambiguity in perceptual anchoring. *AI Communications*, 21(1):1–14, 2008.

[15] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[16] Lennart Ljung. *System Identification: Theory for the User, 2nd ed.* PTR Prentice Hall, 1999.

[17] Amy Loutfi, Silvia Coradeschi, Marios Daoutis, and Jonas Melchert. Using knowledge representation for perceptual anchoring in a robotic system. *Int'l J. Artificial Intelligence Tools*, 17(5):925–944, 2008.

[18] Object Management Group. The CORBA specification v3.1.

[19] Stuart C. Shapiro and Haythem O. Ismail. Anchoring in a grounded layered architecture with integrated reasoning. *Robotics and Autonomous Systems*, 43(2–3):97–108, 2003.

[20] L. Steels. Shared grounding of event descriptions by autonomous robots. *Robotics and Autonomous Systems*, 43(2–3):163–173, 2003.