Multi-Sensor Information Fusion by Query Refinement

Shi-Kuo Chang[1] and Erland Jungert[2]

[1] Department of Computer Science
University of Pittsburgh -chang@cs.pitt.edu
[2] Swedish Defense Research Agency (FOI) – jungert@lin.foi.se

Abstract: In recent years the fusion of multimedia information from multiple real-time sources and databases has become increasingly important because of its practical significance in many application areas such as telemedicine, community networks for crime prevention, health care, emergency management, e-learning, digital library, and field computing for scientific exploration. To support the retrieval and fusion of multimedia information from multiple real-time sources and databases, a novel approach for sensor-based query processing is described. Since most sensors can generate large quantities of spatial information within short periods of time, sensor-based query processing requires new techniques for query optimization. The sensor dependency tree is used to facilitate query optimization. Through query refinement one or more sensor may provide feedback information to the other sensors. The approach is also applicable to evolutionary queries that change in time and/or space, depending upon the temporal/spatial coordinates of the query originator. It provides significant improvements in the accuracy and efficiency in multi-sensor information fusion and accomplishes sensor data independence through the construction of an ontological knowledge base.

1. Sensor-based Query Processing for Information Fusion

In recent years the fusion of multimedia information from multiple real-time sources and databases has become increasingly important because of its practical significance in many application areas such as telemedicine, community networks for crime prevention, health care, emergency management, e-learning, digital library, and field computing for scientific exploration. Information fusion is the integration of information from multiple sources and databases in multiple modalities and located in multiple spatial and temporal domains. Generally speaking, the objectives of information fusion are: a) to detect certain significant events [Waltz90, White98], and b) to verify the consistency of detected events [Chong99, Klein93, Parker99].

As an example, Figure 1(a) is a laser radar image of a parking lot with a moving vehicle (encircled). The laser radar is manufactured by SAAB Dynamics in Sweden. It generates image elements from a laser beam that is split into short pulses by a rotating mirror. The laser pulses are transmitted to the ground in a scanning movement, and when reflected back to the platform on the helicopter a receiver collects the returning pulses that are stored and analyzed. The results are points with x, y, z coordinates and time t. The resolution is about 0.3 m. In Figure 1(a) the only moving vehicle is in the lower right part of the image with a north-south orientation, while all other vehicles have east-west orientation.

Figure 1(b) are two video frames showing a moving white vehicle (encircled) while entering a parking lot in the middle of the upper left frame, and between some of the parked vehicles in the lower right frame. Moving objects can be detected from the video sequence [Jungert99c]. On the other hand, the approximate 3D shape of an object or the terrain can be obtained from the laser radar image [Elmqvist01]. Therefore the combined analysis of laser radar image and video frame sequence provides better information to detect a certain type of object and/or to verify the consistency of the detected object from both sources.

To accomplish the objectives of information fusion, novel sensor-based query processing techniques to retrieve and fuse information from multiple sources are needed. In sensor-based query processing, the queries are applied to both stored databases and real-time sources that include different type of sensors. Since most sensors can generate large quantities of spatial information within short periods of time, sensor-based query processing requires query optimization.

We describe a novel approach for sensor-based query processing and query optimization using the sensor dependency tree. Another aspect to consider is that queries may involve data from more than one sensor. In our approach, one or more sensor may provide feedback information to the other sensors through query refinement. The status information such as position, time and certainty can be incorporated in multi-level views and formulated as constraints in the refined query. In order to accomplish sensor data independence, an ontological knowledge base is employed.

(a) (b)

Figure 1. (a) A laser radar image of a parking lot with a moving vehicle (encircled). (b) Two video frames showing a moving white vehicle (encircled) while entering a parking lot.

There is an important class of queries that require more sophisticated query refinement. We will call this class of queries *evolutionary queries*. An evolutionary query is a query that may change in time and/or space. For example when an emergency management worker moves around in a disaster area, a predefined query can be executed repeatedly to evaluate the surrounding area to find out objects of threat. Depending upon the position of the person or agent (the *query originator*) and the time of the day, the query can be quite different. Our approach is also applicable to evolutionary queries that may be modified, depending upon the temporal/spatial coordinates of the query originator.

This paper is organized as follows. The background and related research are described in Section 2. The notion of sensor data dependence is discussed in Section 3, and the sensor data dependency tree is introduced in Section 4. Section 4 describes simple query processing, and Section 5 illustrated the query refinement approach. Section 6 and Section 7 discuss view management and the sensor data ontological knowledge base, respectively. The σ-join-query examples are presented in Section 8. AN empirical study is described in Section 9. Section 10 discusses the advantages of the proposed research and future research topics.

2. Background and Related Research

In our previous research, a spatial/temporal query language called ΣQL was developed to support the retrieval and fusion of multimedia information from real-time sources and databases [Chang98,

Chang99, Chang2000c, Jungert2000]. ΣQL allows a user to specify powerful spatial/temporal queries for both multimedia data sources and multimedia databases, thus eliminating the need to write separate queries for each. ΣQL can be seen as a tool for handling spatial/temporal information for sensor-based information fusion, because most sensors generate spatial information in a temporal sequential manner [Jungert99a]. A powerful visual user interface called the Sentient Map allows the user to formulate spatial/temporal σ-queries using gestures [Chang2000a, Chang2000b].

For empirical study we collaborated with the Swedish Defense Research Agency who has collected information from different type of sensors, including laser radar, infrared video (similar to video but generated at 60 frames/sec), and CCD digital camera. In our preliminary analysis, when we applied ΣQL to the fusion of the above described sensor data, we discovered that in the fusion process data from a single sensor yields poor results in object recognition. For instance, the target object may be partially hidden by an occluding object such as a tree, rendering certain type of sensors ineffective.

Object recognition can be significantly improved, if a *refined query* is generated to obtain information from another type of sensor, while allowing the target being partially hidden. In other words, one (or more) sensor may serve as a guide to the other sensors by providing status information such as position, time and accuracy, which can be incorporated in multiple *views* and formulated as constraints in the refined query. In the refined query, the source(s) can be changed, and additional constraints can be included in the where-clause of the σ-query. This approach provides better object recognition results because the refined query can improve the result from the various sensor data that will also lead to a better result in the fusion process. A refined query may also send a request for new data and thus lead to a feedback process.

In early research on query modification, queries are modified to deal with integrity constraints [Stonebraker75]. In query augmentation, queries are augmented by adding constraints to speed up query processing [Grafe93]. In query refinement [Velez97] multiple term queries are refined by dynamically combining pre-computed suggestions for single term queries. Recently query refinement technique was applied to content-based retrieval from multimedia databases [Chakrabarti2000]. In our approach, the refined queries are created to deal with the lack of information from a certain source or sources, and therefore not only the constraints can be changed, but also the source(s). This approach has not been considered previously in database query optimization, because usually the sources are assumed to provide the complete information needed by the queries.

In addition to the related approaches in query augmentation, there is also recent research work in agent-based techniques that are relevant to our approach. Many mobile agent systems have been developed [Baumann98, Baumer99, Lange99], and recently mobile agent technology is beginning to be applied to information retrieval from multimedia databases [Kosch2001]. It is conceivable that sensors can be handled by different agents that exchange information and cooperate with each other to achieve information fusion. However, mobile agents are highly domain-specific and depend on ad-hoc, 'hardwired' programs to implement them. In contrast, our approach offers a theoretical framework for query optimization and is applicable to different type of sensors, thus achieving sensor data independence.

3. Sensor Data Independence

As mentioned in the previous sections, sensor data independence is an important new concept in sensor-based query processing. In database design, data independence was first introduced in order to allow modifications of the physical databases without affecting the application programs [Ullman88]. It was a very powerful innovation in information technology. The main purpose was to simplify the use of the databases from an end-user's perspective while at the same time allow a more flexible administration of the databases themselves [Date95].

In sensor-based information systems [Waltz90], no similar concept has yet been suggested, due to the fact that this area is still less mature with respect to the design and development of information systems integrated with databases in which sensor data are stored. Another reason is that the users are supposed to be domain experts and consequently they have not yet requested sensor-based information systems with this property.

In current sensor-based information systems, in order to formulate queries concerning various objects and their attributes registered by the sensors, detailed knowledge about the sensors is required. Therefore sensor selection is left to the users who supposedly are also experts on sensors. However in real life this is not always the case. A user cannot be an expert on all sensors and all sensor data types. Therefore systems with the ability to hide this kind of low-level information from the users need to be developed. User interfaces also need to be designed to allow the users to formulate queries with ease and to request information at a high-level of abstraction to accomplish sensor data independence. An approach to overcome these problems and to accomplish sensor data independence is described, through the use of the sensor dependency tree, the query refinement technique, the multi-level view databases, and above all an ontological knowledge base for the sensors and objects to be sensed.

4. The Sensor Dependency Tree

In database theory, query optimization is usually formulated with respect to a query execution plan where the nodes represent the various database operations to be performed [Jarke84]. The query execution plan can then be transformed in various ways to optimize query processing with respect to certain cost functions. In sensor-based query processing, a concept similar to the query execution plan is introduced. It is called the *sensor dependency tree*, which is a tree in which each node P_i has the following parameters:

$object_i$ is the object type to be recognized
$source_i$ is the information source
$recog_i$ is the object recognition algorithm to be applied
sqo_i is the spatial coordinates of the query originator
tqo_i is the temporal coordinates of the query originator
aoi_i is the spatial area-of-interest for object recognition
ioi_i is the temporal interval-of-interest for object recognition
$time_i$ is the estimated computation time in some unit such as seconds
$range_i$ is the range of certainty in applying the recognition algorithm,
represented by two numbers min, max from the closed interval [0,1]

These parameters provide detailed information on a computation step to be carried out in sensor-based query processing. The query originator is the person/agent who issues a query. For evolutionary queries, the spatial/temporal coordinates of the query originator are required. For other type of queries, these parameters are optional.

If the computation results of a node P_1 are the required input to another node P_2, there is a directed arc from P_1 to P_2. The directed arcs originate from the leave nodes and terminate at the root node. The leave nodes of the tree are the information sources such as laser radar, infrared camera, CCD camera and so on. They have parameters such as (none, LR, NONE, sqo_i, tqo_i, aoi_{all}, ioi_{all}, 0, (1,1)). Sometimes we represent such leave nodes by their symbolic names such as LR, IR, CCD, etc.

The intermediate nodes of the tree are the objects to be recognized. For example, suppose the object type is 'truck'. An intermediate node may have parameters (truck, LR, recog315, sqo_i, tqo_i, aoi_{all}, ioi_{all}, 10, (0.3, 0.5)).

The root node of the tree is the result of information fusion, for example, a node with parameters (truck, ALL, fusion7, sqo_i, tqo_i, aoi_{all}, ioi_{all}, 2000, (0,1)) where the parameter ALL indicates that

information is drawn from all the sources. In what follows, the spatial/temporal coordinates sqo_i and tqo_i for the query originator, the all-inclusive area-of-interest aoi_{all} and the all-inclusive interval-of-interest ioi_{all} will be omitted for the sake of brevity, so that the examples are easier to read.

Query processing is accomplished by the repeated computation and updates of the sensor dependency tree. During each iteration, one or more nodes are selected for computation. The selected nodes must not be dependent on any other nodes. After the computation, one ore more nodes are removed from the sensor dependency tree. The process then iterates. As an example, by analyzing the initial query, the following sensor dependency tree is constructed:

(none, LR, NONE, 0, (1,1)) → (truck, LR, recog315, 10, (0.3, 0.5)) →
(none, IR, NONE, 0, (1,1)) → (truck, IR, recog144, 2000, (0.5, 0.7)) → (truck, ALL, fusion7, 2000, (0,1))
(none, CCD, NONE, 0, (1,1)) → (truck, CCD, recog11, 100, (0, 1)) →

This means the information is from the three sources - laser radar, infrared camera and CCD camera - and the information will be fused for recognizing the object type 'truck'.

Next, we select some of the nodes to compute. For instance, all the three leaf nodes can be selected, meaning information will be gathered from all three sources. After this computation, the processed nodes are dropped and the following updated sensor dependency tree is obtained:

(truck, LR, recog315, 10, (0.3, 0.5)) →
(truck, IR, recog144, 2000, (0.5, 0.7)) → (truck, ALL, fusion7, 2000, (0,1))
(truck, CCD, recog11, 100, (0, 1)) →

We can then select the next node(s) to compute. Since LR has the smallest estimated computation time, it is selected and recognition algorithm 315 is applied. The updated sensor dependency tree is:

(truck, IR, recog144, 2000, (0.5, 0.7)) →
(truck, ALL, fusion7, 2000, (0,1))
(truck, CCD, recog11, 100, (0, 1)) →

In the updated tree, the LR node has been removed. We can now select the CCD node and, after its removal, select the IR node.

(truck, IR, recog144, 2000, (0.5, 0.7)) → (truck, ALL, fusion7, 2000, (0,1))

Finally, the fusion node is selected.

(truck, ALL, fusion7, 2000, (0,1))

After the fusion operation, there are no unprocessed (i.e., unselected) nodes, and query processing terminates.

5. Query Refinement

In the previous section, a straightforward approach of sensor-based query processing is described. This straightforward approach misses the opportunity of utilizing incomplete and imprecise knowledge gained during query processing.

Let us re-examine the above scenario. After LR is selected and recognition algorithm 315 applied, suppose the result of recognition is not very good, and only some partially occluded large objects are recognized. If we follow the original approach, the reduced sensor dependency tree becomes:

(truck, IR, recog144, 2000, (0.5, 0.7)) →
(truck, ALL, fusion7, 2000, (0,1))
(truck, CCD, recog11, 100, (0, 1)) →

But this misses the opportunity of utilizing the incomplete and imprecise knowledge gained by recognition algorithm 315.

If the query is to find un-occluded objects and the sensor reports only an occluded object, then the query processor is unable to continue unless we modify the query to find occluded objects. Therefore a better approach is to refine the original query, so that the updated sensor dependency tree becomes:

(truck, IR, recog144, aoi-23, 2000, (0.6, 0.8)) →(truck, ALL, fusion7, aoi-23, 2000, (0, 1))

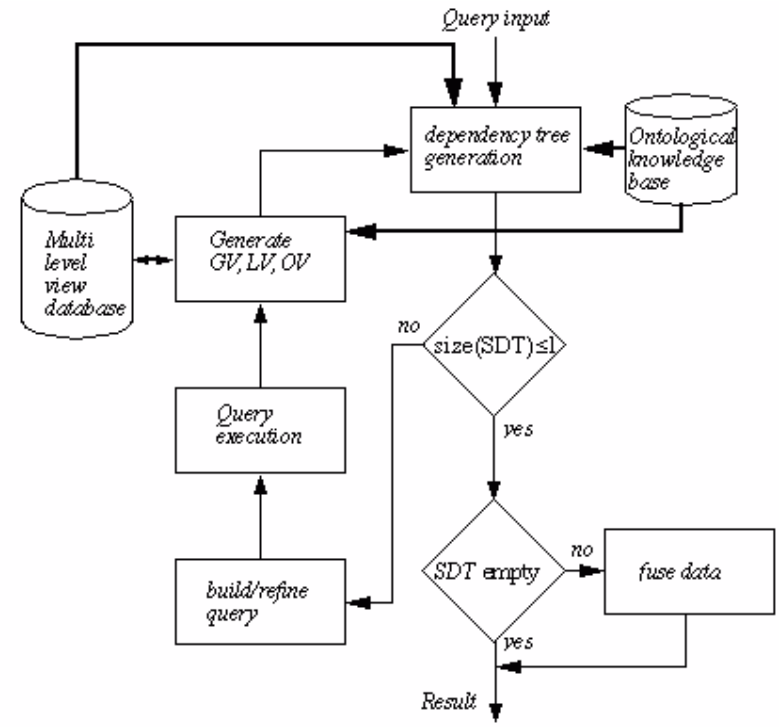

Figure 2. Flowchart for the query refinement algorithm.

This means recognition algorithm 315 is applied to detect objects in an area-of-interest aoi-23. After this is done, the recognition algorithm 144 is applied to recognize objects of the type 'truck' in this specific area-of-interest. Finally, the fusion algorithm fusion7 is applied.

Given a user query in a high-level language, the natural language, a visual language or a form, the query refinement approach is outlined below, where *italic words* indicate operations for the second (and subsequent) iteration. Its flowchart is illustrated in Figure 2.

Step 1. Analyze the user query to generate/*update* the sensor dependency tree based upon the ontological knowledge base and the multi-level view database that contains up-to-date contextual information in the object view, local view and global view, respectively.

Step 2. If the sensor dependency tree is reduced to a single node, perform fusion operation (if multiple sensors have been used) and then terminate query processing. Otherwise build/*refine* the ó-query based upon the user query, the sensor dependency tree and the multi-level view database.

Step 3. Execute the portion of the ó-query that is executable according to the sensor dependency tree.

Step 4. Update the multi-level view database and go back to Step 1.

As mentioned above, there is another class of queries that require more sophisticated query refinement. An evolutionary query is a query that change in time and/or space. Depending upon the position of the *query originator* and the time of the day, the query can be different. In other words, queries and query processing are affected by the spatial/temporal relations among the query originator, the sensors and the sensed objects.

In query processing/refinement, the spatial/temporal relations must be taken into consideration in the construction/update of the sensor dependency tree. The temporal relations include "followed by", "preceded by", and so on. The spatial relations include the usual spatial relations, and special ones such as "occluded by", and so on [Lee92]. As mentioned above, if in the original query we are interested only in finding un-occluded objects, then the query processor must report failure when only an occluded object is found. If, however, the query is refined to "find both un-occluded and occluded objects", then the query processor can still continue.

6. Multi-Level View Database

A multi-level view database (MLVD) is needed to support sensor-based query processing. The status information is obtained from the sensors, which includes object type, position, orientation, time, accuracy and so on. The positions of the query originator and the sensors may also change. This is processed and integrated into the multi-level view database. Whenever the query processor needs some information, it asks the view manager. The view manager also shields the rest of the system from the details of managing sensory data, thus achieving sensory data independence.

The multiple views may include the following three views in a resolution pyramid structure: the global view, the local view and the object view. The *global view* describes where the target object is situated in relation to some other objects, e.g. a road from a map. This will enable the sensor analysis program to find the location of the target object with greater accuracy and thus make a better analysis. The *local view* provides the information such as the target object is partially hidden. The local view can be described, for example, in terms of Symbolic Projection [Chang96], or other representations. Finally, there is also a need for a *symbolic object description*. The views may include information about the query originator and can be used later on in other important tasks such as in situation analysis.

The multi-level views are managed by the view manager, which can be regarded as an agent, or as middleware, depending upon the system architecture. The global view is obtained primarily from the geographic information system (GIS). The local view and object view are more detailed descriptions of local areas and objects. The results of query processing, and the movements of the query originator, may both lead to the updating of all three views.

7. The Ontological Knowledge Base

For any single sensor the sensed data usually does not fully describe an object, otherwise there will be no need to utilize other sensors. In the general case the system should be able to detect that some sensors are not giving the complete view of the scene and automatically select those sensors that can

help the most in providing more information to describe the whole scene. In order to do so the system should have a collection of facts and conditions, which constitute the working knowledge about the real world and the sensors. This knowledge is stored in the *ontological knowledge base*, whose content includes object knowledge structure, sensor and sensor data control knowledge.

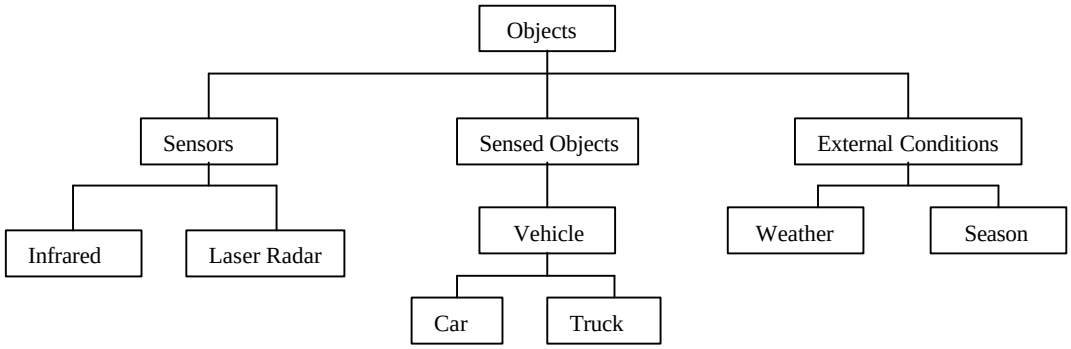

Figure 3. The ontological knowledge base.

An example of the ontological knowledge base is shown in Figure 3. It consists of three parts: the *sensor part* describing the sensors, recognition algorithms and so on, the *external conditions part* providing a description of external conditions such as weather condition, light condition and so on, and the *sensed objects part* describing objects to be sensed. Given the external condition and the object to be sensed, we can determine what sensor(s) and recognition algorithm(s) may be applied. For example, IR and Laser can be used at night (time condition), while CCD cannot be used. IR probably can be used in foggy weather, but Laser and CCD cannot be used (weather condition). However, such determination is often uncertain. Therefore certainty factors should be associated with items in the ontological knowledge base to deal with the uncertainty.

8. Query Examples

An example of a σ-query is illustrated in Figure 4 [Chang98, Chang99]. The *source* R consists of time slices of 2D frames. To extract three pre-determined time slices from the source R, the query in mathematical notation is: $\sigma_t (t_1 , t_2 , t_3) R$.

The meaning of the σ−operator in the above query is “select”, i.e. we want to select the time axis and three slices along this axis. The subscript t in σ_t indicates the selection of the time axis. In the SQL-like language the ΣQL query is expressed as:

SELECT t
CLUSTER t_1, t_2, t_3
FROM R

A new keyword "CLUSTER" is introduced, so that the parameters for the σ−operator can be listed, such as t_1, t_2, t_3. The word "CLUSTER" indicates that objects belonging to the same cluster must share some common characteristics (such as having the same time parameter value). A cluster may have a sub-structure specified in another (recursive) query. Clustering is a natural concept when dealing with spatial/temporal objects. To facilitate query formulation, an alias (the name of a variable) can be given to a cluster.

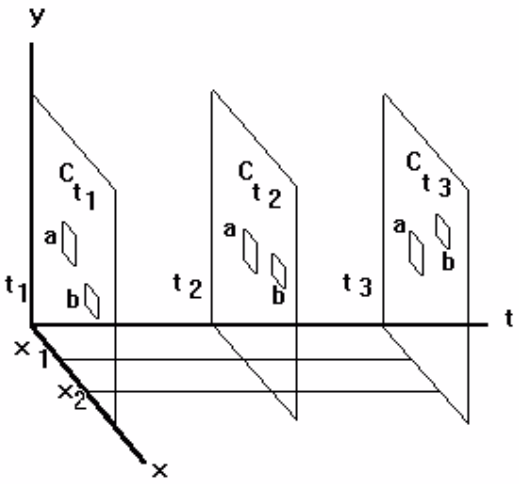

Figure 4. Example of extracting three time slices (frames) from a video source.

In what follows we present three examples to illustrate the processing of queries that involve one or more sensors, and the possibility for query optimization.

Query 1: "Find two frames from a video frame sequence where a frame containing a white bird comes after a frame containing a red bird."

Using the query refinement approach, the steps are as follows.

Step 1. Generate the sensor dependency tree: (none, video, NONE, 0, (1,1)) (white bird, recog192, 20, (0.5,1)). Since the source is given, the sensor dependency tree is a chain.

Step 2. Build the query.

```
// Extract all the frame pairs OBJ1 and OBJ2 such that OBJ1 precedes OBJ2
// in the video sequence and OBJ1 contains an object that is a  red bird
// and OBJ2 contains an object that is a white bird

 SELECT object
 CLUSTER * ALIAS OBJ1 OBJ2
 FROM
        {
                                // it extracts all the video frames
                                SELECT t
                                CLUSTER *
                                FROM video_source
        }
 WHERE OBJ1.type="bird" AND OBJ1.color="red"
 AND OBJ2.type = "bird" AND OBJ2.color="white"
 AND   OBJ1.t < OBJ2.t
```

Note: The clusters of the time axis must be open while the clusters on the object axis must be closed in order to keep the integrity of the frame.

Step 3. Execute the query. Since source is known, we can execute the whole query to obtain the results.

Step 4. Update the multi-level view database.

Step 1 (2nd Iteration). Update the sensor dependency tree. Since it is empty and only one sensor was used, terminate.

Query 2: "Find cars in a region five hours before it is covered by a flood."

Step 1. *Generate the sensor dependency tree*. Initially the source can be any source. By checking the ontological knowledge base and the views, the appropriate source(s) is identified, such as video source, camera, and so on. Once the source(s) has been identified, the sensor dependency tree can be constructed. For this example, there is only one source – the video source.

Step 2. *Build the query*.

```
// Extract all the pairs OBJ1 and OBJ2 such that OBJ1 precedes OBJ2 by 5 hours
// in the frame sequence and OBJ1 contains at least a car and OBJ2 is the starting frame
// for a flood.  Then extract all cars from OBJ1.

SELECT object
FROM
{
  SELECT object
  CLUSTER * ALIAS OBJ1 OBJ2
  FROM
        {
                                // it extracts all the frames
                                SELECT t
                                CLUSTER *
                                FROM video_source
        }
  WHERE OBJ1.type= 'car' AND OBJ2.type = 'starting_flood' AND
  OBJ1.t = OBJ2.t – 5 AND OBJ1.position inside OBJ2.region
}
WHERE object.type= 'car'
```

In the above query, the clusters of the time axis must be open while the clusters on the object axis must be closed in order to keep the integrity of the frame.

Step 3. *Execute the query*. Since source is determined to be a video source, we can execute the query to obtain the results.

Step 4. *Update the multi-level view database*.

Step 1 (2nd Iteration). *Update the sensor dependency tree*. Since the sensor dependency tree is empty and only one sensor was used, we terminate query processing.

Query 3: Same as query 2 but using two sources: a video and a flood meter. This is an example of multi-sensor query processing.

Step 1. *Generate the sensor dependency tree*. By checking the ontological knowledge base and the views, the source for car information is determined to be the video source, and the source for flood information is determined to be the flood meter.

Step 2. Build the query.

```
SELECT object
CLUSTER *
FROM
        {
                // For each frame in the following cluster, the query extracts all the objects
                // and declares an alias for these objects
          UNION(
                SELECT object
                CLUSTER * ALIAS OBJ1
                FROM
                        {
                                SELECT t
                                CLUSTER *
                                FROM video_source
                        }

                SELECT object
                CLUSTER * ALIAS OBJ2
                FROM
                        {
                                SELECT t
                                CLUSTER *
                                FROM flood_meter
                        }
                )
        }
        WHERE OBJ1.type='car' AND OBJ2.type = 'starting_flood'
        AND OBJ1.t = OBJ2.t - 5 AND OBJ1.position inside OBJ2.region
}
WHERE object.type= 'car'
```

Step 3. Execute the query. Since both sources are known, we can execute the query to obtain the results.

Step 4. Update the multi-level view database.

Step 1 (2nd Iteration). Update the sensor dependency tree. Since it is reduced to a single node, and two sensors were used, we can perform fusion operation and then terminate query processing.

It can be seen that the processing of the second query can be improved, if the flood meter is processed first, so that the flood region is first determined, before the processing of video frames to identify cars.

9. An Experimental Study

For empirical study we collected over 700 G-bytes of data from three type of sensors, including laser-radar, infrared camera (similar to video but generated at 60 frames/sec), and a CCD digital camera. Figure 5 is an example of an infrared image, a laser radar image and a CCD image of the same area.

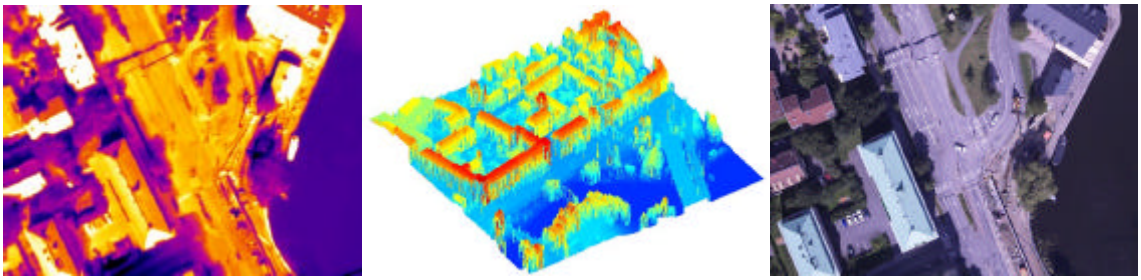

Figure 5. An infrared image(left), a laser radar image (middle) and a CCD image (right).

This experimental data is provided by the Swedish Defense Research Agency for the evaluation of the sensor-based query processing approach. High resolution terrain elevation models for synthetic environments are produced using laser-radar data [Jungert99b]. GIS data are also available, so that the multi-level view databases and the ontological knowledge base can be constructed.

Researchers at the Swedish Defense Research Agency plan to collect a substantial number of test queries. Three different types of test queries will be of interest: a) queries for the recognition of objects from multiple sensors; b) spatial/temporal queries; and c) evolutionary queries. Using these test queries and the experimental data, we will be able to conduct an experimental study.

10. Discussion

This paper provides a general and formal framework for formulation and optimization of sensor-based queries, which has three major advantages: a) the ability to utilize incomplete information in sensor-based query processing and optimization, b) the generality to deal with spatial/temporal queries, and c) the achievement of sensor data independence. Given this framework, here are many research topics for further investigation.

10.1. Query Optimization

From the discussion in previous sections, it can be seen that query refinement needs to take into consideration the following factors:

Sources may change: For instance, if we process a query during the day time, certain type of sensors are applicable. But if we run the same query at night, certain type of sensors may become inapplicable. In this case the time of the day and the ontological knowledge base determine the applicability of queries.

Constraints may change: The processing of a partial query for some sensor input may lead to a downsized area-of-interest or shorter time-interval-of-interest.

Clusters may change: The processing of a partial query may lead to the change of clusters, for example, the change of resolution in the video frames. Due to a higher priority task, the sampling time for the frames may change.

Query originator may change: If the query is an evolutionary query, the spatial and temporal coordinates of the query originator may change, leading to changes in the spatial/temporal relations among query originator, sensors and sensed objects.

After a query is processed, changes in the views lead to a refined query and an updated sensor dependency tree. The refinement of the query must be done with respect to the ontological knowledge base and the views. Since one main objective of query refinement is to optimize query processing, the

cost function for query processing need to be specified, and optimization algorithms can then be investigated.

8.2. Specification and Processing of Evolutionary Queries

As described in previous sections, some queries such as the evolutionary queries need to be processed repeatedly, each time with minor changes, during a certain period of time and/or within a certain geographical area. This can be specified through predefined queries.

Since most users of sensor-based information systems are not experts in sensors, we can attack the problem in two ways. By constructing an ontological knowledge base, the low-level detailed information about sensors, objects to be sensed and environmental conditions can be stored in the ontological knowledge base. By providing query templates or a sentient map interface, the commonly encountered queries can be specified by form filling or active map navigation. Indeed, the σ-join-queries have the following standard form:

select-cluster-join

They correspond to the select-project-join SQL join queries [Date95]. To formulate such σ-join-queries, a template-filling approach can be used. Dimensions, sources, join conditions can all be based upon selections from pull-down menus. In the WHERE part, if the type of an object is set to 'aaa', there must be an algorithm to recognize the object 'aaa', or the object 'aaa' is already in the database. This can be determined by checking the ontological knowledge base.

To support evolutionary queries, a further refinement of the technique is to let user specify queries with additional parameters, such as how often to run the query, in what time intervals and geographical areas, and so on. Since evolutionary queries usually change incrementally, we need to investigate techniques to dynamically optimize evolutionary query processing, by maximizing the sharing of information in the processing of consecutive evolutionary queries.

8.3. Type-M Dependency Relations

We need to further investigate how to employ type-M dependency relations in expressing sensor data dependence in the ontological knowledge base. Type-M dependency relation is a new type of dependency relation to characterize the dependency in multimedia databases [Polese2001]. A function f: X---s_1,s_2--->Y is a type-M dependency relation if and only if f: X--->Y is a dependency relation, s_1 is a similarity relation in the domain X, and s_2 is a similarity relation in the domain Y. Since X and Y can also be domains for sensor data, we need to further investigate extending the notion of type-M dependency relation to sensor data. This will facilitate the construction of sensor dependency tree.

8.4. Uncertainty Management in Query Processing and Optimization

As mentioned above, certainty factors should be associated with the nodes in the sensor dependency tree, the items in the ontological knowledge base, as well as data acquired by the sensors due to technical imperfections in the sensors and other practical considerations. Certainty factors (or confidence values) are normalized as real numbers in the interval [0,1] and interpreted as the certainty (or the confidence) a user may have in a query result. Therefore in query processing, all the computation steps should take uncertainty management into consideration. Different approaches in uncertainty management, including Bayesian networks [Jensen96] and fuzzy logic [LLNL2001], can be considered. Since the certainty factor of a node may change after a computation step, there may be multiple ways of deciding the precedence among the nodes, and the sensor dependency tree may be replaced by the *sensor dependency graph*. Query processing then does not proceed by simply eliminating nodes successively from the sensor dependency graph. We need to investigate the

generalized solution, such as using relaxation algorithms, to attack the problem of query processing and optimization with uncertainty management.

References:

[Baumann98] J. Baumann et al., "Mole – Concepts of a Mobile Agent System", *World Wide Web*, Vol. 1, No. 3, 1998, pp 123-137.

[Baumer99] C. Baumer, "Grasshopper – A Universal Agent Platform based on MASIF and FIPA Standards", First International Workshop on Mobile Agents for Telecommunication Applications (MATA'99), Ottawa, Canada, October 1999, World Scientific, pp 1 -18.

[Chakrabarti2000] K. Chakrabarti, K. Porkaew and S. Mehrotra, "Efficient Query Refinement in Multimedia Databases, 16th International Conference on Data Engineering, San Diego, California, February 28 – March 3, 2000.

[Chang96] S. K. Chang and E. Jungert, *Symbolic Projection for Image Information Retrieval and Spatial Reasoning*, Academic Press, London, 1996.

[Chang98] S. K. Chang and E. Jungert, "A Spatial/temporal query language for multiple data sources in a heterogeneous information system environment", *The International Journal of Cooperative Information Systems* (IJCIS), vol. 7, Nos 2 & 3, 1998, pp 167-186.

[Chang99] S. K. Chang, G. Costagliola and E. Jungert, "Querying Multimedia Data Sources and Databases", Proceedings of the 3rd International Conference on Visual Information Systems (Visual'99), Amsterdam, The Netherlands, June 2-4, 1999.

[Chang2000a] S. K. Chang, "The Sentient Map", *Journal of Visual Languages and Computing*, Vol. 11, No. 4, August 2000, pp 455-474.

[Chang2000b] S. K. Chang, T. H. Chen and C. S. Li, "Gesture-Enhanced Information Retrieval and Presentation in a Distributed Learning Environment", Proceedings of the International Conference on Multimedia (ICME'2000), New York, July 31 to August 2, 2000.

[Chang2000c] S. K. Chang, G. Costagliola and E. Jungert, "Spatial/Tem poral Query Processing for Information Fusion Applications", Proceedings of the 4th International Conference on Visual Information Systems (Visual'2000), Lyon, France, November 2000, Lecture Notes in Computer Sciences 1929, Robert Laurini (Ed.), Springer, Berlin, pp 127-139.

[Chong99] C.-Y. Chong, S. Mori, K.-C Chang and W. H. Baker, "Architectures and Algorithms for Track Association and Fusion", Proceedings of Fusion'99, Sunnyvale, CA, July 6 -8, 1999, pp 239-246.

[Date95] C. Date, *An Introduction to Database Systems*, Addison-Wesley, 1995.

[Elmqvist01] M. Elmqvist, E. Jungert et al., "Terrain Modelling and Analysis using Laser Scanner Data", Proceedings of Conference on Land Surface Mapping and Characterization using Laser Altmetry, Annapolis, MD, USA, October 22-24, 2001, pp 219-226, published by Dept. of Geography, University of Maryland, MD, 2001.

[Grafe93] G. Grafe, "Query Evaluation Techniques for Large Databases", *ACM Computing Surveys*, Vol. 25, No. 2, June 1993.

[Jarke84] M. Jarke and J. Cohen, "Query Optimization in Database Systems", *ACM Computing Surveys*, Vol. 16, No. 2, 1984.

[Jensen96] F. V. Jensen, *An Introduction to Bayesian Networks*, Springer Verlag, New York, 1996.

[Jungert99a] E. Jungert, "An Information fusion System for Object Class ification and Decision Support Using Multiple Heterogeneous Data Sources", Proceedings of the 2nd International Conference on Information Fusion (Fusion'99), Sunnyvale, California, USA, July 6-8, 1999.

[Jungert99b] E. Jungert, U. S[illegible]erman, S. Ahlberg, P. H[illegible]ling, F. Lantz, G. Neider, "Generation of high resolution terrain elevation models for synthetic environments using laser-radar data", Proceedings of SPIE no 3694, Modeling, Simulation and Visualization for Real And Virtual Environments, Orlando , Florida, April 7-8, 1999, pp 12-20.

[Jungert99c] E. Jungert,"A Qualitative Approach to Reasoning about Objects in Motion Based on Symbolic Projection", Proceedings of the Conference on Multimedia Databases and Image Communication (MDIC'99), Salerno, Italy, October 4-5, 1999.

[Jungert2000] E. Jungert, "A Data Fusion Concept for a Query Language for Multiple Data Sources", Proceedings of the 3rd International Conference on Information Fusion (FUSION 2000), Paris, France, July 10-13, 2000.

[Klein93] L. A. Klein, "A Boolean Algebra Approach to Multiple Sensor Voting Fusion", *IEEE Transactions on Aerospace and Electronic Sys*tems, Vol. 29, No. 2, April 1993, pp 317-327.

[Kosch2001] H. Kosch, M. Doller and L. Boszormenyi, "Content -based Indexing and Retrieval supported by Mobile Agent Technology", *Multimedia Databases and Image Communication*, LNCS2184, (M. Tucci, ed.), Springer-Verlag, Berlin, 2001, pp 152-166.

[Lange99] D. B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley, Reading, MA, USA, 1999.

[LLNL2001] Lawrence Livermore National Laboratory, "Multisensor data fusion system using fuzzy logic", in the web site on sensor technology at http://www.llnl.gov/sensor_technology/ STR25.html, 2001.

[Lee92] S.Y. Lee and F. J. Hsu, "Spatial Reasoning and Similarity Retrieval of images using 2D C -string knowledge Representation", *Pattern Recognition*, vol. 25, no 3, 1992, pp 305-318.

[Paquet98] Eric Paquet and Marc Rioux, "A Content -based Search Engine for VRML Database", Proceedings of 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp 541-546, 1998.

[Parker99] J. R. Parker, "Multiple Sensors, Voting Methods and Target Value Analysis", Proceedings of SPIE Conference on Signal Processing, Sensor Fusion and Target Recognition VI, SPIE vol. 3720, Orlando, Florida, April 1999, pp 330-335.

[Polese 2001] G. Polese and S.K. Chang, "Towards a Theory of Normalization for Multimedia Databases", Proceedings of IEEE Conference on Human Centered Computi ng HCC2001, Stresa, Italy, September 2001.

[Stonebraker75] M. Stonebraker, "Implementation of Integrity Constraints and Views by Query Modification", in SIGMOD, 1975.

[Ullman88] J. D. Ullman, *Database and Knowledge-base Systems*, Vol 1, Computer science Press, Rockville, Maryland, USA, 1988, pp 11-12.

[Velez97] Bienvenido V帾ez, Ron Wiess, Mark A. Sheldon, and David K. Gifford, "Fast and Effective Query Refinement", Proceedings of the 20th ACM Conference on Research and Development in Information Retrieval (SIGIR97), Philadelphia, Pennsylvania, July 1997.

[Waltz90] E. Waltz and J. Llinas, *Multisensor data fusion*, Artect House, Boston, 1990.

[White98] F. E. White, "Managing Data Fusion Systems in Joint and Coalition Warfare", Proceedings of EuroFusion98 – International Conference on Data Fusion, October 1998, Great Malvern, United Kingdom, pp 49-52.