Proceedings of the 6th AGILE
April 24th-26th, 2003 – Lyon, France

MULTIRESOLUTION FOREST INFORMATION: USING HARVESTER AS A FOREST SURVEY TOOL AND HOW TO PROCESS AND STORE THE DATA

Timo Melkas, Jussi Rasinmäki

Department of Forest Resource Management, University of Helsinki,
P.O.Box 27, 00014 University of Helsinki, Finland
e-mail: timo.melkas@helsinki.fi, jussi.rasinmaki@helsinki.fi

1. INTRODUCTION

In the recent years there has been a strong drive globally for efficient monitoring of the use of natural resources. Forest is one of the most important natural resources, especially so on the boreal vegetation zone. In Finland alone over 55 million m³ of timber is cut annually as raw material for the forest industry [1].

Effective monitoring requires information on the resources. Depending on the use of forest resource information in monitoring, inventory and planning, there are several resolutions at which the information should be produced. Some forestry operation planning tasks utilise very detailed data whereas for regional level strategic planning coarser level is sufficient. In this paper we present a novel approach to obtain forest resource information in several resolutions simultaneously.

Most of the logging in Finland is done with harvesters, which are forest machines designed to fell, delimb and cut trees to specified log lengths and pile the logs in the forest. The GSM wireless network connects the 1 500 harvesters in Finland with the saw- and paper mills. This enables the real time transfer of desired log dimensions from the mills to the controlling system of the harvester. While the cut tree is being moved through the processing head the measuring unit of the harvester can store log diameters every ten centimetres, length of the log and several other variables [2]. Most of the modern harvesters also have GPS (Global Positioning System) equipment. The driver uses a map display of GPS-positions as an aid to monitor the location of the harvester in relation to the borders of the harvested stand and possible sensitive areas to be avoided.

Combining the GPS observations with the detailed log measurement information produces new kind of forest resource information. Using located tree and log level data produced by a harvester we aimed to develop methods to process and store multiresolution, hierarchical forest resource data. We created methods to assign trees to an arbitrary region inside the stand so that information of the tree composition and volume could be produced at required resolution. On the highest, stand level, resolution we tested a method to estimate the boundaries of the stand. To be usable in operational activities in forestry and monitoring, the data processing methods are not sufficient by themselves. Effective methods to store and access the multiresolution data are also needed. We evaluated the ability of two conceptually different data management solutions to store and provide access to the data on different levels of resolution.

2. MATERIAL AND METHODS

2.1 Material

The material consisted of three different stands in Southern Finland. The logging in each case was a final felling i.e. all trees were cut, and the sizes of the stands ranged from 0.5 to 2 ha. The tree species composition ranged from pure common spruce and Scots pine stands to a mixture of pine, spruce and silver birch. From each cut tree the measuring system of the harvester recorded the diameter at breast height, the length of the utilisable stem part and the stem diameter every ten centimetres. The harvester cut each tree into one or several logs and recorded their length, top diameter and volume. Using the coordinate data from a GPS system with a roof mounted antenna, the measurement system of the harvester combined the harvester position data and the tree data at the moment the tree was cut (fig. 1). The location data included also information on the quality of GPS signal (number of satellites, hdop, time since DGPS signal).

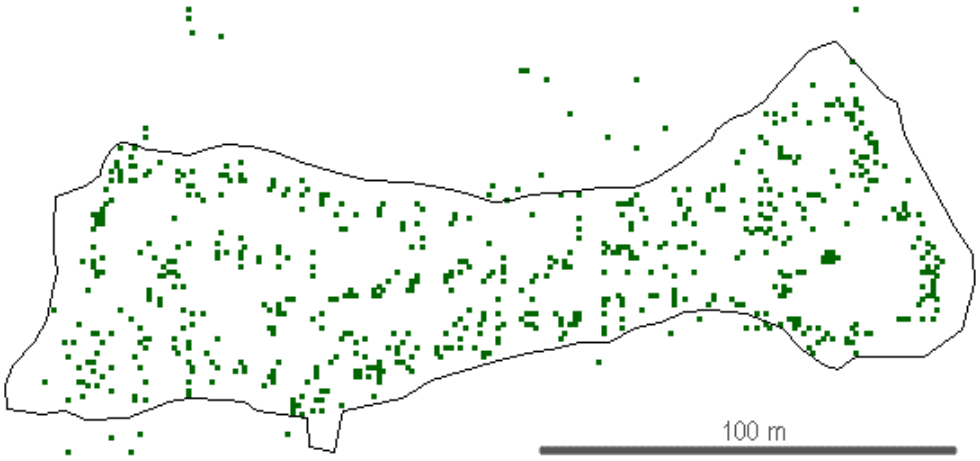


Fig. 1 A clearcut stand and the harvester locations

We collected separate location material by measuring the coordinates of the points where the harvester was at the time each tree was cut, and the location of all cut trees in the stands. The harvester locations were measured with a GPS capable of submetre accuracy and the tree locations from a set reference points using laser distance and angle measurement devices. We measured also the exact location of stand border by walking along the stand perimeter with a GPS receiver. The perimeter was deemed to be halfway between the outmost cut trees and trees left standing.

The data set for data model comparison consisted of data collected from over 200 stands with five different harvesters in Southern Finland between 2000 and 2002.

2.2 Data processing

We assigned trees to the subregions of the stand using a concept of uncertainty zone of a tree. The uncertainty zone reflects the fact that the locations recorded for trees are in fact locations for the harvester. This means that the real location of the tree can be inside the circle which has the radius of the reach of the harvester's processing head.

Because of the uncertainty in the tree location, we used Monte Carlo simulation when determining which trees belong to which subregions. In the simulation the location of each tree was sampled from a tree location probability distribution. After the simulation we

calculated the probability of a tree belonging to a region as a ratio between those realizations where the tree was in the region and the total number of realizations.

Two factors determined the location probability distribution for a tree: the probability of a tree to be cut (i) at a given distance from the harvester and (ii) at a given angle to the direction the harvester was moving. We used the manually located material to estimate these probability density functions.

We based the estimate of the distance probability distribution on the frequency histogram of the distances between the harvester and the trees logged from each harvester location (fig. 2).

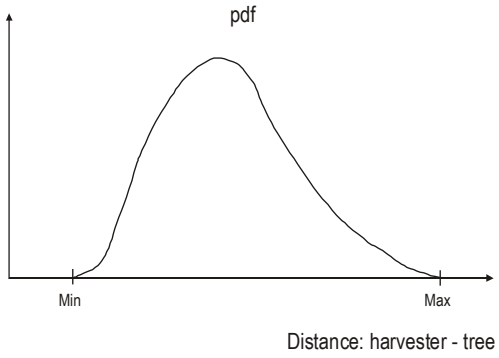


Fig. 2 A schematic representation of a possible probability density function for the distance between the harvester and the cut tree. The size of the harvester sets the minimum distance and the reach of the processing head the maximum distance.

When determining the angle probability density function we took three locations—previous, current and next—and grouped these settings into classes 'harvester moving straight on' and 'harvester turning' to see whether the probability density functions would differ.

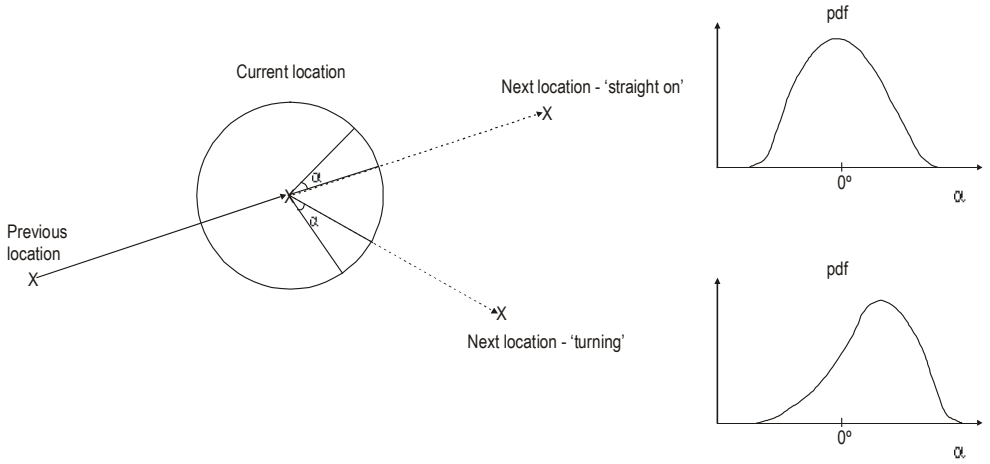


Fig. 3 Schematic representations of possible probability density functions for the angle between the direction the harvester is moving and the location of the cut tree.

After the simulation we calculated the total volume for each region as

$$V_{region} = \sum_{i=1}^{n} v_i \cdot p_i$$

where v_i is the volume of the tree i and p_i is the probability that the tree is inside the region.

The effect of the accuracy of the original location observation was assessed by doing the simulation twice: once using the original harvester location observations and once using the manually located harvester positions.

We used bootstrapping to calculate the confidence intervals for the estimates of total volumes of the regions. These results were then compared to the result obtained using the manually measured tree location data to assess the accuracy of volume estimate.

The top level in the hierarchy of data produced from the harvester logged data was the stand level and the objective was to delineate the harvested stand. We did this by buffering all of the observed harvester locations with the distance equal to the reach of the harvester. Again we used Monte Carlo simulation to quantify the effect of GPS-location inaccuracy on the results of buffering. Before the simulation we excluded any clear outlier observations using the original delineation of the stand, which had been used as a prior information for the harvesting operation. We sampled a new location in each simulation run from a uniform distribution covering a circle. The radius of the circle was sampled from a normal distribution with zero mean and standard deviation equal to that of GPS-location recordings. The original measured location was the centre of the sampling circle. We did the buffering on the sampled locations and overlaid the results to get a fuzzy delineation of the stand: the more often a point falls inside the buffered region the higher is its degree of membership in the stand. Out of these overlaid regions a single delineation for the stand can be drawn by a rule 'a point belongs to the stand if it is inside the buffered regions x times out of hundred'.

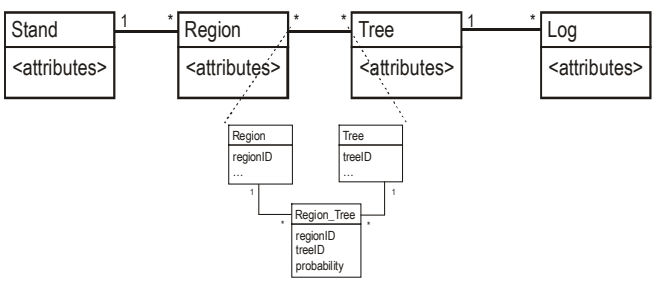
We compared the simulated stand border and the independently measured one for area and location match.

2.3 Data storage

We tested two different database based data management solutions for their performance statistics when accessing the hierarchical forest data that was produced as explained above. We attempted to maximize the conceptual difference between the models. One was based on describing the structure of the data as closely as possible already at the conceptual model level whereas the other was designed to be abstract in its definition at the conceptual level. This leaves the definition of the actual contents of the data to the data entry level.

The first option used basic relational data model [3] where the data is normalised into the 4th normal form: separate tables with well defined sets of attributes (fig. 4b). The conceptual data model for this option is presented in fig. 4a using UML-notation [4]. The hierarchical nature of data was reflected by 1:N and M:N-relationships between the tables.

a)



b)

LOG	
logID	NUMBER
treeID	NUMBER
WGS_location	GEOMETRY
cutTime	NUMBER
fixQuality	NUMBER
satellites	NUMBER
hdop	NUMBER
altitude	NUMBER
geoidHeight	NUMBER
timeSinceDGPS	NUMBER
lenght	NUMBER
topDiameter	NUMBER
volume	NUMBER
buttOffset	NUMBER
assortmentName	STRING
assortmentNumber	STRING
qualityName	STRING
qualityNumber	NUMBER
forcedCut	NUMBER

Fig. 4 a) The basic relational data model, implementation of the M:N relationship between regions and trees. b) An example of an attribute set of a single table

The second option used a more abstract conceptual data model. The model itself only specified the top level abstract feature class and its generic structure (fig. 5a). This model used the object-oriented extensions to the relational model as defined in the ISO/IEC 9075-n:1999 standard set also known as SQL3 or SQL:1999. The hierarchical structure of the data—log, tree, region and stand—was implemented using the object-relational pointer-like concept of reference to other object instance. The references form a graph structure (fig. 5b). A feature has superfeatures on the higher hierarchical level and subfeatures on the lower level: trees are associated to logs as subfeatures and to subregions as superfeatures, likewise trees are subfeatures and stands superfeatures to regions. As described below for feature attributes, nested tables stored these references for each feature instance. A single database table was used store features from all hierarchy levels as the association with the semantics class resolved the meaning of any single feature.

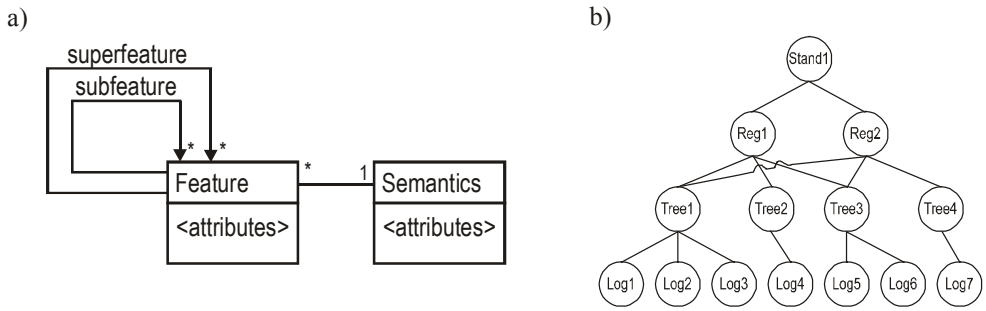
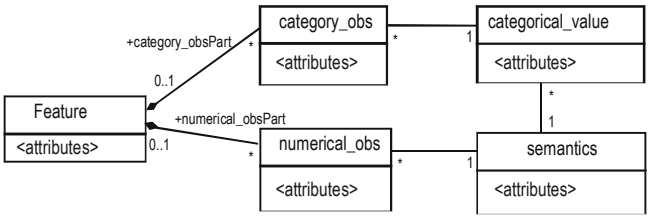


Fig. 5 Conceptual data model based on abstract feature class, semantics class and hierarchy associations. b) An example of the hierarchy graph that the super- and subfeature associations define.

The semantics class also took care of the specification of the attributes of different features. There were only two generic attributes for features: numerical observations and categorical observations (fig. 6a). The number of attributes for a single feature, however, was unlimited as both of the generic attributes are declared to be of type nested table. The nested relational data model [5] removes the first normal form restriction of basic relational model by accepting multivalued attributes (fig. 6b). With every attribute value also a reference to the semantics class was stored to give meaning to the value.

Categorical and numerical feature attributes were separated to satisfy the requirements of the second normal form. Categorical attributes were stored in the category_obs part of the feature class. Each instance of category_obs referenced the categorical_value class which enumerated the possible values of each categorical variable. Again the categorical_value instances referenced the semantics class for the meaning of each categorical variable. The structure of numerical attributes was more straightforward. Numerical_obs had a value and a reference to the semantics class to give a meaning for the value.

a)



b)

Feature	
ID	Numerical_obs
1	<DBH, 20> <volume, 250>
2	<height, 5> <DBH, 3> <volume, 13>
3	<volume, 20>

Fig. 6 Object-relational features of the abstract model: nesting of feature attributes using multivalued attributes of abstract data type. b) An example table illustrating the nested table concept. For clarity, the meanings of values given as text instead of references to the semantics class.

We implemented the conceptual models on Oracle 9*i* database management system and compared the performance of the different conceptual models for different relational algebra operations. We optimised each query using both cost and function based optimisers and recorded the fastest execution time.

The relational algebra operations tested were SELECT, PROJECT, JOIN and two AGGREGATE FUNCTIONS.

SELECT: selection of rows from a relation – all attributes of a tree with certain conditions.

SQL-implementations:

Relational model:

```
SELECT *
 FROM TREES
 WHERE DBH>20
```

Object-relational model:

```
SELECT f.id, DEREF(n.semantics).name, n.value
 FROM FEATURES f, TABLE(f.numerical_obs) n
 WHERE f.id IN
  (SELECT s.id
    FROM FEATURES s, TABLE(s.numerical_obs) u
    WHERE DEREF(u.semantics).name='DBH' AND u.value>20)
UNION
SELECT f.id, DEREF(DEREF(c.value).semantics).name, DEREF(c.value).value
 FROM FEATURES f, TABLE(f.categorical_obs) c
 WHERE f.id IN
  (SELECT s.id
    FROM FEATURES s, TABLE(s.numerical_obs) u
    WHERE DEREF(u.semantics).name='DBH' AND u.value>20)
```

For the object-relational implementation the selection of all attributes under certain conditions translates into a union of nested queries. Nested query structure is necessary as only one attribute per row is stored in the nested attribute tables. Therefore a query is first needed to find out the feature ids for which the condition of the main query is satisfied. Union operation is needed because the numerical and categorical attributes are stored in separate nested tables.

The object-relational operators used in the query are: DEREF carries out the dereferencing of the object instance reference i.e. it returns the instance that the reference points to. After dereferencing the standard object-oriented dot notation can be used to access the attributes of the object instance. TABLE flattens the nested table so that it can be queried as a standard table that is joined to its parent table

PROJECT: selection of columns from a relation – certain attributes for all trees.

Relational model:

```
SELECT tree_id, species, dbh
 FROM TREES
```

Object-relational model:

```
SELECT f.id, DEREF(c.value).value
 FROM FEATURES f, TABLE(f.categorical_obs) c
 WHERE DEREF(DEREF(c.value).semantics).name='species'
  AND DEREF(f.semantics).name='tree'
UNION
SELECT f.id, n.value
 FROM FEATURES f, TABLE(f.numerical_obs) n
 WHERE DEREF(c.semantics).name='DBH'
  AND DEREF(f.semantics).name='tree'
```

For PROJECT operation again a union is needed in the case that the projection contains both numerical and categorical attributes. The attribute list is in the WHERE-part of the query statement whereas for the basic relational query it is given in the SELECT-part. This reflects the fact that for relational model the attribute set for each feature type is fixed and for object-relational model it is unspecified at database design time.

JOIN: combinations of tuples from both relations that satisfy the join conditions – all trees for a region inside the stand.

Relational model:

```
SELECT t.tree_id
 FROM TREES t, REGIONS r
 WHERE  r.region_id='REGION1' AND t.region_id=r.region_id
```

Object-relational model:

```
SELECT DEREF(t.feature_ref).id
 FROM FEATURES f, TABLE(f.subfeatures) t
 WHERE f.id='REGION1' AND DEREF(f.semantics).name='region'
```

AGGREGATE FUNCTIONS: sum, average

Aggregate function inside one table – average volume of logs

Relational model:

```
SELECT AVERAGE(volume)
 FROM LOGS
```

Object-relational model:

```
SELECT AVERAGE(n.value)
 FROM FEATURES f, TABLE(f.numerical_obs) n
 WHERE DEREF(n.semantics).name='volume'
  AND DEREF(f.semantics).name='log'
```

Aggregate function involving a join – volume of trees in a region

Relational model:

```
SELECT SUM(t.volume * rt.probability)
 FROM REGION_TREE rt, TREES t
 WHERE  rt.region_id='REGION1' AND t.tree_id=rt.tree_id
```

Object-relational model:

```
SELECT SUM(n.value * sf.probability)
 FROM features f, TABLE(f.numerical_obs) n, TABLE(f.superfeature) sf
 WHERE f.id IN
  (SELECT DEREF(l.feature).id
    FROM features g, TABLE(g.subfeature) l
    WHERE g.id='REGION1')
  AND DEREF(sf.feature).id='REGION1' AND DEREF(n.semantics).name='volume'
```

If the aggregate function query involves a join between hierarchy levels, a nested query is again needed. The subquery returns the ids for features from the other level that are associated with the higher level feature. That information can then be used in the WHERE-clause in the similar manner as in the relational query.

Results and discussion on the data processing methods and on the comparison between two data management models will be presented at the conference.

3. REFERENCES

[1] Finnish Statistical Yearbook of Forestry 2001. Forest Statistics Information Service, Finnish Forest Research Institute, Helsinki, Finland.

[2] Skogforsk, 2002. StanForD. Http://www.skogforsk.se/marknad/stanford/estart.htm. November 2002.

[3] Codd, E. F., 1970. A Relational Model of Data for Large Shared Data Banks. CACM 13(6): 377-387.

[4] Rumbaugh, J., Jacobson, I. and Booch, G., 1998. The Unified Modeling Language Reference Manual. - Addison-Wesley.

[5] Makinouchi, A., 1977. A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model. In: Proceedings of the Third International Conference on Very Large Data Bases, October 6-8, 1977, Tokyo, Japan, pp. 447-453.