Magic Revisited: The Object-Oriented Solution to a Cadastral Maintenance Problem

Walter Hesse PhD (Melbourne)
Major Projects Group
Melbourne Information Technology Services (MITS)

and

Ian P. Williamson PhD(NSW), LS, MISA
Professor and Head of Department
Department of Surveying and Land Information
The University of Melbourne

Abstract

This paper has evolved from the ongoing research in the area of "Optimising, Maintaining and Updating the Spatial Accuracy of Digital Cadastral Data Bases", a paper published earlier in The Australian Surveyor [Hesse et al. 1990]. This paper gives additional background information for the process described in the first paper and describes new findings in form of an object-oriented software implementation. The concepts of this relatively new software development approach are examined and their advantages for cadastral modelling and software creation are demonstrated. The resultant software prototype, programmed in Smalltalk/V286, has been implemented and tested at the Department of Surveying and Land Information, The University of Melbourne. Future trends are discussed with special emphasis on the rapidly changing hardware and software platforms, their impact on cadastral and LIS issues and the challenges ahead for the continuing education of Land Information professionals.

Introduction

As found by [Hesse et al.1990],

...the potential of a DCDB as a base map for a land information system is, in part, restricted by its graphical accuracy and depends largely on the methods and tools used to initially capture this information.

This view was confirmed by a Review of Australasian Digital Cadastral Data Bases (DCDB) by [Hesse and Williamson 1990a] and many requests by utility users of the DCDB, to address these issues with quality research. The research is (only) concerned with the spatial accuracy in relation to a geodetic reference framework rather than with completeness or validity of the data (see [Hunter 1991b] for a detailed discussion of this important area).

Where the earlier paper justified the use of a certain adjustment technique, the so called 'MAGIC' approach, and found that

...The method can be used to maintain this accuracy once achieved and further used as a method to update existing data bases of inferior accuracy with more accurate data sets, a situation typical for the subdivisional process....

the authors also pointed to further improvements in the methodology, although significant accuracy improvements had been achieved. These improvements are presented in this paper, furthermore additional information regarding the adjustment techniques and formulas are supplied, following many requests by interested readers of The Australian Surveyor.
Background

The basic concept of MAGIC, or Multiple Adjustment of Geometry with Implicit Constraints, is based on an adjustment technique first described by [Schmid 1965], and tested with good results in a German cadastral system by [Haag 1986]. The method was tested and compared with other methods used in digital mapping [Hesse 1986]. A suggestion by [Boljen 1987] to take additional area constraints into account forms now part of the MAGIC program. The original VAX-IGDS program has been converted to a UNIX-MicroStation version at the Department of Surveying and Land Information at The University of Melbourne. The adjustment treats graphical points (boundary corners) as observations in a 'network' made up from parcel boundaries (distance observations). These line observations can also be part of other 'pseudo observations' in the form of orthogonalities, straight lines, parallelisms or area observations, see figure 1.

![Figure 1 - Pseudo - Observations](image1)

The observation equation for e.g. a 90 degree observation is:

\[ l = 90.000^\circ \]

\[ v = \arctan \left( \frac{y_2-y_0}{x_2-x_0} \right) - \arctan \left( \frac{y_1-y_0}{x_1-x_0} \right) + \frac{\pi}{2} \]

First partial derivative:

\[ v = a \cdot dx_0 + b \cdot dy_0 + c \cdot dx_1 + d \cdot dy_1 + e \cdot dx_2 + f \cdot dy_2 - dl \]

\[ a = \frac{y_2-y_0}{(D_{02})^2} - \frac{y_1-y_0}{(D_{01})^2} \]

\[ b = \frac{x_1-x_0}{(D_{01})^2} - \frac{x_2-x_0}{(D_{02})^2} \]

\[ c = \frac{y_1-y_0}{(D_{01})^2} \]

\[ d = -\frac{x_1-x_0}{(D_{01})^2} \]

\[ e = -\frac{y_2-y_0}{(D_{02})^2} \]

\[ f = \frac{x_2-x_0}{(D_{02})^2} \]

\[ dl = \arctan \left( \frac{y_1-y_0}{x_1-x_0} \right) - \arctan \left( \frac{y_2-y_0}{x_2-x_0} \right) + \frac{\pi}{2} \]

The differentials for straight lines, parallels, distances, areas, etc... are obtained in a similar way. These pseudo observations (or implied constraints) can be derived with the help of the error propagation law, estimating the error of the graphical point coordinates. The unknowns in this model are the (point-) coordinate shifts dX and dY. By introducing weights for all observations (including the original digitised points) the system is user controllable and flexible and matrix singularity will become impossible. The methodology becomes much clearer by using the following example:

![Figure 2 - Adjustment Example](image2)

12 coordinate (digitised) observations (points 1-6):

Obs. 1 - 12 : (X,Y)_1, (X,Y)_2, (X,Y)_3, ....(X,Y)_6

3 Angle (pseudo) observations:

Obs. 13: angle 1-2-5 = f ( (X,Y)_1, (X,Y)_2, (X,Y)_5 ) =!= 180°
Obs. 14: angle 4-3-6 = f ( (X,Y)_4, (X,Y)_3, (X,Y)_6 ) != 180°

Obs. 15: angle 4-1-2 = f ( (X,Y)_4, (X,Y)_1, (X,Y)_2 ) != 90°

1 Length (pseudo) observation:
Obs. 16: frontage 1-2 = f ( (X,Y)_1, (X,Y)_2 ) != 5.45 m

2 Area (pseudo) observations:
Obs. 17: area 1-2-3-4 = f ( (X,Y)_1, .. (X,Y)_4 ) != 150 sqm
Obs. 18: area 2-5-6-3 = f ( (X,Y)_2, .. (X,Y)_3 ) != 150 sqm

This would result in a configuration matrix A of:

\[
\begin{array}{cccccccccccc}
\text{dX1} & \text{dY1} & \text{dX2} & \text{dY2} & \text{dX3} & \text{dY3} & \text{dX4} & \text{dY4} & \text{dX5} & \text{dY5} & \text{dX6} & \text{dY6} & \text{P} \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
3 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
5 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
6 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
7 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
8 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
9 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
12 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
13 & * & * & * & * & * & * & * & * & * & 1000 & 1000 & 1000 \\
14 & * & * & * & * & * & * & * & * & * & 1000 & 1000 & 1000 \\
15 & * & * & * & * & * & * & * & * & * & 1000 & 1000 & 1000 \\
16 & * & * & * & * & * & * & * & * & * & 1000 & 1000 & 1000 \\
17 & * & * & * & * & * & * & * & * & * & 1000 & 1000 & 1000 \\
18 & * & * & * & * & * & * & * & * & * & 1000 & 1000 & 1000 \\
\end{array}
\]

Table 1 - A-Matrix and Weight vector P

The '*' in table 1 symbolizes values as calculated above, the weight values in P are arbitrary, user definable and dependent on the source or reliability of the observation, point (-coordinate) observations get the lowest weight.

**Desired Improvements**

Overall, the MAGIC approach allows to

- significantly enhance the data quality of existing graphical data bases;
- maintain this quality;
- facilitate further upgrades; and
- integrate non-cadastral layers of information, e.g. utility data.

Although good improvements of the spatial accuracy of the digitised data were achieved, further improvements to the methodology were desired in terms of a better user interface and data quality visualization during the adjustment process. Large data volumes are still a solvable problem, this will have to be addressed with the use of fast hardware and by appropriate algorithms such as [Kruck 1983] or the work done by [Clatworthy 1991]. The real problems were seen in the data structure and software creation field. The separation of geometry in form of the graphics files and the storage of additional
constraints in a separate neutral file format, as done in the first MAGIC attempt, was not the ideal situation.

The desired 'on-line adjustment' with automatic presentation of data quality, a better communication of possible constraint violations during update cycle and a better integrated graphical user interface all lead to the investigation of object oriented programming languages and modelling techniques, explained with the example of an object-oriented MAGIC solution in the following section. In this form of combined data and procedural storage, the data primitives or objects that 'make up' the cadastre should 'know' their dependencies and relations to the surrounding cadastre and react to changes made to neighbouring data (e.g. moving to a new position or signaling the violation of possible constraints). The following section will highlight the differences and demonstrate a considerable improvement in overall functionality.

**An Object-Oriented Software Solution**

While the conventional solution described above is operational, there were a number of shortcomings:

- The software becomes difficult to extend
- The visualization of error, or data quality in general, was difficult.
- Data editing in the conventional vector structure became clumsy.
- The overall user interface was poor.

Similar observations were made by [Pullar 1989] who, in his paper on data organization of a digital cadastral data base, expresses concerns

> *...that current commercial GIS do not support some of the operational aspects of a digital cadastre ... providing alternative to data linkages between coverage layers is not a difficult task from a software engineering perspective for a GIS. It is the fact that this functionality is absent from commercial GIS's that introduces the problem.*

Although the object-oriented TIGRIS software modules were available throughout the studies, the TIGRIS programming interface for the standard user licence is restricted to the use of the Parametric Programming Language (PPL) which allows only for a streamlining of standard (vendor supplied) functionality, but not for the creation of new (user-defined) object structures. This resulted in the purchase of Digitalk's Smalltalk/V286 PC package. The Smalltalk environment allowed the creation of a complete new data structure for the modelling of the digitised cadastre and the internal constraints, without being restricted to any fixed data structure such as CAD graphics files or the relational data base model. Furthermore the WIDGETS Graphical User Interface classes [Acumen Software 1990], based on Smalltalk/V286, allowed for the development of a unique user interface which was tailored specifically for the MAGIC cadastral maintenance problem. Figure 3 gives an overview of the class structure and hierarchy used for the MAGIC/Smalltalk version.

**Figure 3 - MAGIC/Smalltalk Class Hierarchy**

The object hierarchy in this particular MAGIC/Smalltalk version of the software is grouped in five main categories, all subclassed to the highest level *Object*. The main MAGIC functionality is contained in subclasses of *ApplicationWindow* and *MagicObs*. The classes *Network* and *Node* where re-used from existing Smalltalk classes and modified or subclassed (with *MagicNode*) to accommodate a certain behaviour needed for the MAGIC application. Category "Other" contains objects such as numbers, text panes, class browsers, etc. - the full explanation of these being outside the scope of this thesis.

Figure 3 shows a conceptual overview of the existing and self-developed classes. A far better
appreciation of the overall functionality of the MAGIC/Smalltalk version can be gained by interactively working with the package on the PC. On initiation of the application the user is presented with a number of options in the form of pull-down menus and menu-icons, which are arranged according to a typical cadastral maintenance work-flow session and ordered in a left-to-right and top-down fashion. It has to be noted that the layout of the graphical user interface (GUI) is completely user definable, and can be, with the help of the WIDGET class InterfaceEditor, very quickly re-arranged to suite personal preferences. The icons have the following meaning, as shown in figure 4 and 5:

**Figure 4 - MAGIC Icons**

**Figure 5 - MAGIC Icons**

Once external data have been imported, the icons can be used to activate various ApplicationWindow subclass objects in the form of WeightEditor (for variable weighting of 'pseudo-observation' during adjustment); StatisticDisplay (to keep track of data volumes), DisplayChooser (for variable graphical presentation of digitised data and additional constraints), ColorChooser ( for data quality visualization); or Analyzer (for the computer-assisted training and automatic detection of 'pseudo' constraints).

The ListBox object "Points" lists the point numbers that make up the cadastral parcel pattern, ListBox"Obs" shows all current 'pseudo' observations and ListBox "Dep" list the points which depend on the observation selected (highlighted) in the "Obs" ListBox . When a point or observation is selected from the list boxes the selected object is highlighted graphically, the amount of highlight-blink rate can be defined by the user through the DisplayChooser. The graphical selection of points with the mouse cursor results in the highlighting of the appropriate point in the ListBox , the 'dragging' of points updates the easting and northing values in their boxes and editing of the coordinate value boxes results in the automatic graphical shift of the point. The graphical symbolization of points and observations is entirely user definable. Data quality of points is typically defined with "green" for 'good' (accurate), "pink" for 'questionable' quality and "red" for 'problem cases'. The Analyzer object class manages the detection and storage of the so called 'pseudo' constraints. Upon activation the user is presented with a statistical distribution plot of the various observation types (area, azimuth, distances ...) to assist users while they are 'searching' for possible instances of 'implicit' constraint.

The push of the "Adjust" icon performs a least square adjustment of all observations, outside of the Smalltalk environment, using already existing FORTRAN matrix operation subroutines for the processing of the adjustment. The adjustment algorithm could have been programmed entirely within the Smalltalk, or any other object-oriented, environment. The advantages of this possible approach in general would have been program design consistency and data integrity. The disadvantages of this possible approach would have been the time loss involved in re-coding already functional (FORTRAN) code and the (relatively) slow performance rate for matrix operations within the Smalltalk environment, due to the fact that the language has to interpreted at run-time. Already existing (FORTRAN or any other high-level language or assembler) code can easily be incorporated into the Smalltalk environment by calling so called 'user primitives'. In the case of the latest Smalltalk/Windows release this is facilitated by accessing dynamic link libraries (DLL) or dynamic data exchange (DDE) which is completely integrated with Windows 3.0 on the PC [Digitalk 1991].

A justification for time-consuming re-coding of conventional code can only be found when the objects (and their encapsulated behaviour) that would have to be designed, could take advantage of the object-oriented strengths of class hierarchy and inheritance. This is not the case for a 'static' adjustment of data, a process clearly defined by the rigours of least square adjustment techniques and the underlying concepts of matrix algebra - a method not changed since its first appearance and use in surveying 150 years ago.
The very nature of matrix operations is fixed by its strict rules and there will probably never be the need 'to subclass' or change the behaviour of a matrix operation as such. The process can be 'hidden' as a user primitive which returns a certain (fast) result, just as the basic operations (methods such as adding, multiplying, dividing) at the lower level of the Smalltalk language are performed by Assembler code primitives.

However, a possible justification for the use of 'matrix objects' can be found for the case of a more 'dynamic' adjustment process, described in a later section on associativity issues between various GIS data layers. The matrices involved in this adjustment process are dependant on the 'observation objects', their weights and their dependant 'point observation objects'. A possible dynamic change of any of these components could be 'triggered' (through message-passing) to the 'configuration matrix object' (or affected parts thereof) which in turn results in a 'normal matrix object' inversion and the display of the newly adjusted 'point objects' together with their appropriate 'error ellipse objects'. Although the adjustment algorithm is the (mathematical) core of the MAGIC program, the real advantages of the object-oriented design are found more in the 'shell' of the program - the possibilities of data abstraction, graphical data manipulation and representation - in short a far better data model and graphical user interface.

After adjustment, the original digitised data is displayed in gray as a 'background', the colour of the now shifted points indicate the amount of shift for visual inspection. Points which have shifted out of the user defined tolerance could be placed in an 'error queue' for further close inspection and editing of surrounding constraints, ready for another adjustment. This cyclic nature of the adjustment process can be varied by use of the WeightEditor or addition/deletion of 'pseudo' constraints. This operation is a mixture of mathematical analytical reasoning and a certain amount of interpretation by the user. The important fact is that the original position of the digitised point is always retained as an attribute (an instance variable) within the MagicNode object instances. This value remains stored together with an estimate of its standard error according to initially estimated, empirically based values for the original digitised point or to mathematically derived estimates from the adjustment process. With this information the points can be displayed with or without error ellipses depending upon the switch in DisplayChooser and therefore guide the operator during maintenance procedures.

A closer look at the overall object hierarchy and the MAGIC Smalltalk code demonstrate how the three most powerful aspects of object-oriented programming (encapsulation, polymorphism and inheritance), were put into good use for this version of the MAGIC application. The specific behaviour of different observation types is encapsulated in the various sub-classes of MagicObs, yet they are safe to be stored together in a 'data container' object of type Dictionary, called ObsData; for example, the "drawUpdate" method of the Magic (ApplicationWindow subclass) object contains the instruction to draw the content of ObsData on a map (i.e. the screen):

ObsData do: [:i | i drawOnWidget: displayMap].

The various observations obviously have to be drawn differently (e.g. distance observations in green with a certain offset to the connecting line between two points, angle observations in red drawn as an arc), but this is hidden from the application programmer whose only interest is in seeing them 'drawn', not worrying how each individual observation type implements this behaviour internally. The different observation types 'react' differently to the same message - this is polymorphism.

Probably the best example for the advantages of inheritance for the MAGIC application is the PenWidget subclass MapWidget. The WIDGET GUI already had the object class PenWidget which allowed the placement of graphic primitives, such as lines, circles and text within its extent. This placement was based on screen coordinates (pixel resolution) with its origin in the upper left corner of the screen. For a more natural mapping of GIS data to the screen, some of these methods had to be redefined in the MapWidget subclass to allow the placement of (real-world) coordinates and variable view scale. By inheriting the majority of the functionality already provided in PenWidget and extending only the functionality desired
specifically for this application (such as a proper map coordinate system, mouse control and data manipulation), very fast development progress was made.

A combination of both polymorphism and inheritance is found in the MagicObs method called "blinkOnWidget" which can be activated for any observation type (by inheritance).

1 to: BlinkRate do: [:i |


self drawOnWidget: aWidget].

The actual processes of observation 'erasing' and 'redrawing' (resulting in 'blinking' on the screen) are implemented at the specific observation level using polymorphism.

The most astonishing aspect of the MAGIC/Smalltalk development was the speed and ease of prototyping of the software. None of the conventional, highly specialized (and expensive!) CAD or GIS packages in the Department seemed to be appropriate to model the cadastral constraints and general functionality described within this chapter. A general purpose object-oriented development package (to the value of $250) and some third party add-on modules (to the value of $150) in the form of the Widgets GUI was used instead to create a package that did exactly what the user had in mind. No compromise was made by adopting a 'foreign' CAD or GIS data structure or 'work-around' solutions. The result is highly functional, visually pleasing, easily expandable in terms of sub classing and 'generally flexible' for ad-hoc changes and future software maintenance. The productivity advantage over conventional software development techniques can be estimated at 5:1 to 10:1 or higher depending on the complexity of the problem. A figure not dissimilar to the generally accepted value for the gain in programming power between Assembler and high level conventional languages is therefore possible.

**Associativity issues**

The extension of the MAGIC functionality to cover associativity issues between LIS data layers (typically between the cadastre and utility organisations) could be achieved easily. The groundwork is already laid through the introduction of the concept of 'pseudo' observations. They could be realized through utility offset measurements to the cadastre if available, or through the automatic creation of 'pseudo' offsets when no measurements are available and when the utility networks have been digitised 'on top' of the cadastre. This creation process could generate the needed 'offsets' for the utility nodes, typically where pipes or cables connect or change direction or attributes, by searching for the nearest surrounding cadastral boundary linework to project the utility node onto the cadastral line, thereby creating 'pseudo' observations in the form of offset measurements. Should the (digitised) cadastral base subsequently shift, the 'attached' utility network would start to shift as well, maintaining relativity to the cadastre.

The ideal case of a constant and immediate 'on-line-adjustment', as soon as any component of the data set is being moved, is not far away. This would only be dependant on the size of the adjustment matrices (i.e. the number of points that form part of the adjustment) and the processing speed of the hardware. Alternatively the system could be configured so that only the closest components would be affected by geometry changes. This could be done through the 'triggering' (through message passing between objects!) of dependant node movements. Earlier studies into an object-oriented Engineering Modelling System (EMS) [Intergraph 1988c] showed that the concepts of associativity are well researched and implemented for mechanical engineering application (moving machine components, etc...). EMS is based on the same graphical core software modules as TIGRIS, but the associativity between features in TIGRIS is not (yet) implemented. The conceptual handling of associativity between a utility layer and a cadastral layer in a LIS has been discussed and simulated for demonstration purposes, through the use of HyperCard by [Hebblethwaites 1989]. A more general case, in the form of a rule-based system for the propagation of constraining information in geometric structures, is presented by [Aldefeld 1988]. [Kühn 1990] describes
the prototype of a system that utilises a geometric constraint calculus, based on a least square adjustment technique similar to the technique described at the beginning of this chapter.

Conclusions

1. The current degree of spatial accuracy in digitised cadastral maps is of concern for some users of digital cadastral data sets, especially for the engineering design needs of the utility industry.
2. Current commercial LIS software is not well suited to model, implement and maintain spatial integrity of multi-layered LIS data sets. The theoretical aspects of the improvements needed are known from overseas experience, but the actual implementation of geometric constraints in the Australian context is new.
3. The MAGIC software development, even in its conventional (i.e. non-object-oriented) form, means a significant improvement of current cadastral data maintenance tasks in Australasia. Test results show an increase in spatial accuracy at a relatively low cost [Hesse et al. 1990].
4. The conventional solution is workable, but restrictive in its extension capabilities and unsatisfactory in terms of data modelling and graphical user interface.
5. The object-oriented solution, based on Smalltalk and the Widgets GUI, showed great improvement in terms of development time, software maintainability, graphical user interface, ease of use and extendability when compared to the conventional software solution.
6. The modelling of associativity between various LIS layers is possible through the introduction of 'pseudo' offset observations and subsequent global adjustment or the local 'triggering' of geometric constraints.

References


Haag, K. 1986, "Realisierung geometrischer Bedingungen bei der Digitalisierung von Katasterkarten", *AVN* Heft 5, 190 - 202


Symposium on Spatial Database Accuracy, 203-214, DoSLI, The University of Melbourne


