Journal of Applied Geodesy 4 (2010), 201–217 © de Gruyter 2010. DOI 10.1515/JAG.2010.020

Application of artificial intelligence in Geodesy – A review of theoretical foundations and practical examples

Alexander Reiterer, Uwe Egly, Tanja Vicovac, Enrico Mai, Shahram Moafipoor, Dorota A. Grejner-Brzezinska and Charles K. Toth

Abstract. Artificial Intelligence (AI) is one of the key technologies in many of today's novel applications. It is used to add knowledge and reasoning to systems. This paper illustrates a review of AI methods including examples of their practical application in Geodesy like data analysis, deformation analysis, navigation, network adjustment, and optimization of complex measurement procedures. We focus on three examples, namely, a geo-risk assessment system supported by a knowledge-base, an intelligent dead reckoning personal navigator, and evolutionary strategies for the determination of Earth gravity field parameters. Some of the authors are members of IAG Sub-Commission 4.2 – Working Group 4.2.3, which has the main goal to study and report on the application of AI in Engineering Geodesy.

Keywords. Artificial Intelligence, Knowledge-Based Systems, Fuzzy Logic, Artificial Neural Networks, evolutionary strategies, Geo-Monitoring, Navigation, Gravity Field Determination.

1. Introduction

AI, in general, is the study and design of intelligent agents, where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success (Russell and Norvig 2003). Many real-world problems require the agent to operate with incomplete or *uncertain information* (Negnevitsky 2005). Methods used for uncertain reasoning are probabilistic in nature, such as Bayesian networks, which represent a general tool that can be used for a large number of problems, for example, reasoning (using the Bayesian inference algorithm) (Cooper 1990), learning (using the expectation-maximization algorithm) (Ghahramani and Rowei 1999), planning (using decision networks) (Subbu and Sanderson 2004), and perception (using dynamic Bayesian networks) (Ferreira et al. 2008). Probabilistic algorithms can also be used for filtering, prediction, smoothing and finding explanations for streams of data, helping perception systems to analyze processes that occur over time. AI techniques also include classifiers and statistical learning methods. A general introduction to AI can be found in Russell and Norvig (2003) and Negnevitsky (2005).

In this paper, we show three different methods of AI and their application in Geodesy. The *first example* shows a decision support system which accesses the risk of a landslide using a knowledge-based system. One of the challenging tasks when building such a system is the transfer of the knowledge from experts to the system. The *second example* demonstrates the application of AI for an intelligent personal navigator supported by a Dead Reckoning (DR) mechanism. In this particular example, DR is achieved by integrating over time the incremental motion expressed by step direction, step length, and step altitude. In this context, the term "intelligent navigation" represents the transition from the conventional point-to-point DR to dynamic navigation using the knowledge about the mechanism of the moving person. This knowledge is implicitly represented by a human locomotion model. The *third example* deals with the determination of Earth gravity field parameters (spherical harmonic coefficients). Usually, this is done by a least-squares approach leading to huge normal equation matrices that have to be inverted. We present an alternative direct method based on evolutionary strategy (ES), which is working without the need to invert any matrix at all – the algorithm makes use of the principles of mutation and selection.

Using AI techniques has many advantages in comparison to conventional development and implementation strategies (Russell and Norvig 2003, Negnevitsky 2005): quick access to the collected knowledge (e.g., knowledge-based systems), easy to implement prototypes without deep expert knowledge (e.g., artificial neuronal networks), or systems that are able to learn (e.g., genetic algorithms). Nevertheless, the development and use of AI-based technologies and methods is often controversially discussed (Marr 1977).

2. Modern techniques of artificial intelligence in geodesy

Methods and techniques of AI are commonly used in Geodesy. In 1995 the research project SAMBA ("*System zur Anwendung der Messtechnik im Bauwesen*") was designed to develop a knowledge-based system for the analysis of bridge deformations (Kuhlmann 1993). A special aspect of the work done by Brezing (2000) is dedicated to the modeling of the knowledge – the system has been implemented from scratch without a knowledge-based system shell. Reiterer (2001) developed a prototype of a motorized digital level including control software and a knowledge-based system for data analysis. Chmelina (2002) presented the concept and prototype of knowledge-based software for the automated detection of significant 3D displacement behavior in tunneling. ANNs and neuro-fuzzy networks have been used by Heine (1999) and Miima (2002) for deformation analysis. The automatic modeling of cause-effect relations for applications in geodetic deformation monitoring has been studied by Martin and Kutterer (2007), Neuner and Kutterer (2010) and Vicovac et al. (2010). Actual research work is dedicated to the modeling and optimization of the efficiency of measurement processes by Petri Nets and Genetic Algorithms (Rehr and Kutterer 2010). Other relevant papers are Wieser (2002), Haberler-Weber et al. (2007), Heinert and Niemeier (2007), Riedel and Heinert (2008). More applications of AI techniques in Geodesy can be found in Reiterer and Egly (2008) and Reiterer et al. (2010).

In the following we will give a brief overview about some of the most relevant techniques of AI for Geodesy:

- Knowledge-Based Systems and Fuzzy Logic Systems,
- Artificial Neural Networks (ANN),
- Evolutionary Algorithms (EA) with its two branches – Genetic Algorithms (GA) and Evolutionary Strategies (ES).

2.1. Knowledge-based systems and fuzzy systems

(a) Knowledge-based systems

In general, Knowledge-Based Systems (KBS) are software systems which solve a non-trivial problem in a specific domain on an expert level (Stefik 1998, Negnevitsky 2005). Typically, the problems considered are hard to solve by analytic methods and the solving methods are often based on rules of thumb and heuristics. Before we turn our attention to a specific implementation methodology we discuss the architecture of a KBS (see Figure 1).

The most important components are the knowledge base consisting of the domain-specific and case-specific knowledge, and the inference component which allows drawing conclusions from the knowledge base. These two components are often called the core of the KBS. The knowledge base may contain knowledge representations in different forms, most often procedural and *declarative. Procedural* knowledge is typically incorporated as a short algorithm or program, whereas *declarative* knowledge is represented explicitly.

Depending on the application context, there may be additional components as well. The most important is the user interface (UI), consisting of the interface for the expert and the interface for the inexperienced user. A knowledge acquisition component is also often available that with the expert edits/updates/changes the knowledge base. In principle, it is even possible that the KBS updates itself by learning, but this is seldom implemented in KBS when applied to technical problems. A further part is the explanation

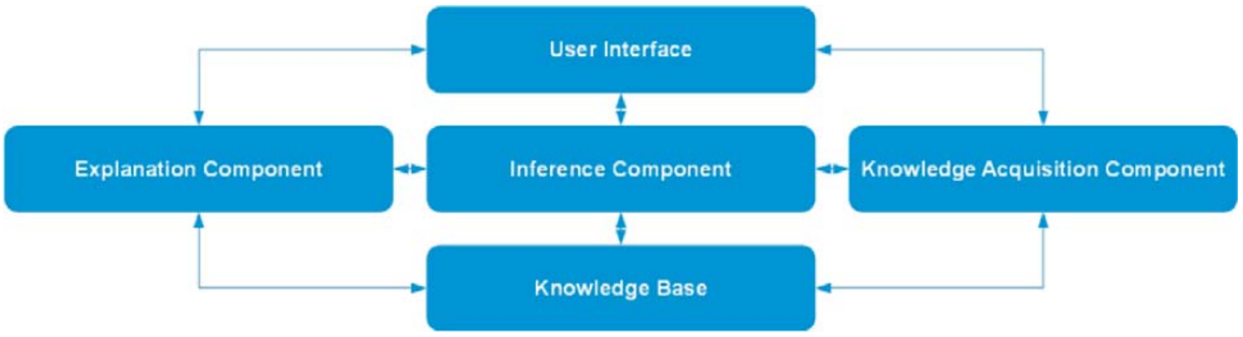

Figure 1: Architecture of a Knowledge-Based System (Stefik 1998).

component that is important for the acceptance of KBS. It collects relevant data during the solving process in order to explain the solution (and the finding of it) to the user. In many cases, the explanation component plays an important role in the testing and evaluation phases of a KBS.

The features and advantages of knowledge-based systems are as follows (Debenham 1989):

- Most of the problem-specific knowledge is represented declaratively. This means that the main focus of developing such a system is on the knowledge which is implemented and not on how to implement it.
- The knowledge is separated from the control strategy. This separation supports re-usability of knowledge and makes an easy manipulation of knowledge possible.
- Symbolic representation of knowledge enables the system to explain the problem-solving process and the solution, which results in an improved user acceptance. Such explanations cannot be achieved with sub-symbolic approaches like neural networks.
- Not only "hard" knowledge can be represented, but also "vague/heuristic/loose" knowledge[1] (which is useful and potentially very profitable).

A knowledge-based approach can deal with uncertainty. Uncertainty occurs, for instance, if one is not absolutely certain about a piece of information. Uncertainty is represented by a numerical value (between 0 and 1), where the value 1 indicates that the user/system is sure that the information is true, and a factor of 0 that he/she/it is unsure (Negnevitsky 2005).

After a more general discussion about KBS, we turn our attention now to a specific KBS model. For small KBS, which are often embedded into technical processes, the rule-based approach (rule-based system – RBS) is beneficial. Moreover, it allows an easy integration of uncertainty.

The philosophy of the rule-based approach is very similar to the way people solve problems. Human experts find it convenient to express their knowledge in the form of rules, which often describe situations in which specific actions have to be performed. Therefore, one possible interpretation of rules (but not the only one) is in terms of situation-action pairs. Furthermore, rules are a way to represent knowledge without complex programming constructs, because we declaratively describe the situation and the action without completely specifying the complete control regime when a rule has to apply.

The core of a rule-based system consists of a fact base for case-specific knowledge, a rule base for domain-specific knowledge, and the recognize-act-cycle (RAC) is the inference component. For our implementation we are using the rule-based programming language JESS (Jess 2010); for the following examples we will adopt this system's syntax.

The fact base of a rule-based system is the *working memory*, i.e., a kind of database which is a collection of *working memory elements*. These elements are instantiations of a *working memory type* and can be considered as record declarations in PASCAL or struct declarations in C. In JESS *working memory types* are templates and realized by the deftemplate operator. This operator provides the knowledge engineer with the possibility to introduce slots (fields) into a fact. These slots can be accessed by names. The deftemplate construct is used to introduce a template or type from which concrete instantiations (or facts) can be derived. The following definition of a template is explained in detail.

```
(deftemplate meteo_local
       (slot nr (type INTEGER))
       (slot id (type STRING))
       (slot date (type INTEGER))
       (slot time (type INTEGER))
       (slot temp_local (type FLOAT))
       (slot temp_local_fuzzy (type SYMBOL)           (1)
             (allowed-symbols very_low
             low middle high very_high))
       (slot precipitation (type FLOAT))
       (slot precipitation_fuzzy (type SYMBOL)
             (allowed-symbols none very_low
             low middle high very_high))
)
```

The template *meteo_local* is a *working memory type* consisting of several *slots*, namely, *nr, id, date, time,*

1 Evolution Strategies (ES) can also incorporate "heuristic" knowledge by applying so-called pre-defined penalty terms to the performance index, i.e., the quality function.

temp_local, temp_local_fuzzy (the fuzzy value of the local temperature), *precipitation* and *precipitation_fuzzy* (the fuzzy value of the local precipitation), respectively.

Associated to each slot is a type restricting possible slot values. Possible types are INTEGER, STRING, FLOAT or SYMBOL. SYMBOL means that a symbol can be stored in the slots. The symbols allowed for each of the slots are defined with "*allowed-symbols*." Type checking is performed during runtime in order to guarantee that the content of a slot satisfies its definition. An example of a JESS fact is shown in the following:

```
(meteo_local
        (nr 1)
        (id 001)
        (date 24042009)
        (time 075545)                              (2)
        (temp_local 14.5)
        (temp_local_fuzzy middle)
        (precipitation 0.0)
        (precipitation_fuzzy none)
)
```

The knowledge base in a rule-based system (often called production systems) represents the domain-specific knowledge and consists of a set of rules (or productions). A rule itself consists of two parts, namely, the *left-hand side* (LHS) and the *right-hand side* (RHS). In the LHS, we formulate the preconditions of the rule; the conjunction of all preconditions describes the situation. In the RHS, the actions are formulated that are applied if the rule is executed or fired. In principal, a rule can be fired if all its preconditions are satisfied. Firing a rule means the execution of the actions specified in the right-hand side. It is important to note that the satisfaction of all preconditions in the left-hand side of the rule does not imply that the rule is immediately fired – it only means that the rule can be fired. The decision which rule to fire in a specific situation is made by the inference or control component.

The inference component or inference engine in a rule-based system is often the recognize-act-cycle, which itself can be controlled by the knowledge engineer in a declarative way (see Figure 2). In a first step, the LHS of all rules are checked against all possible combinations of working memory elements – this exhaustive check is called the *matching-phase*. The result of the matching phase is the *conflict set*, which includes all *rule instances* "ready to be fired". Generally speaking, such a rule instance consists of the rule together with all *working memory elements* that are responsible that the rule's LHS is satisfied. A *conflict resolution strategy* selects one *rule instance* which is actually fired. Such a strategy is often based on a given rule priority (or salience), on specify (prefer a rule which describes a more detailed situation) or on the time the information has been made available to the system (Debenham 1989).

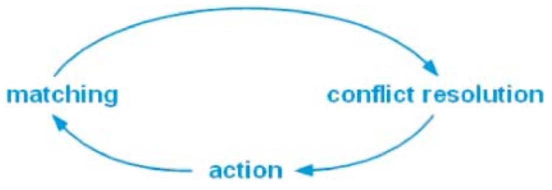

Figure 2: Simplified representation of the recognize-act-cycle.

(b) Fuzzy systems

As mentioned before, a knowledge-based system can be extended by Fuzzy Logic (FL), combining fuzzy sets with fuzzy rules to assess complex non-linear behaviors (Negnevitsky 2005). A fuzzy set A in a non-empty set X is defined by the membership functions η_A, interpreted as the degree of membership of each element x in a fuzzy set A over the unit interval:

$$A = \{x, \eta_A(x)\}, \quad \eta_A : x \to [0,1], \quad x \in X. \quad (3)$$

This means that the value of the membership function indicates the degree of membership of a quantity x in the fuzzy set. If the membership value is 1, the quantity is perfectly representative of the set, and if it is 0, the quantity is not at all a member of the set. Membership functions are usually represented as parametric functions, such as triangular, trapezoidal, or bell-shaped functions. A typical configuration of a fuzzy based system, as shown in Figure 3, has four fundamental components: (1) a fuzzification component (or fuzzifier), (2) a fuzzy rule base, (3) an inference engine, and (4) a defuzzification component.

The main idea of a fuzzy logic algorithm is to use linguistic fuzzy rules to capture vague concepts presented by fuzzy sets. The fuzzification process is the act of transforming the crisp, measured value of an input variable into a linguistic representation (i.e., relating to a fuzzy set), and finding the numerical value of the membership function for that variable.

The lengths of the upper and lower segments of the membership functions are the design parameters, and the choice of the overlap determines the number of rules invoked. The actual shape of the member-

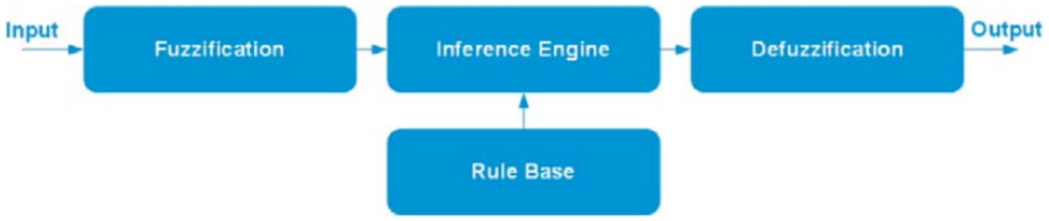

Figure 3: Architecture of a fuzzy based system (Kosko 1991).

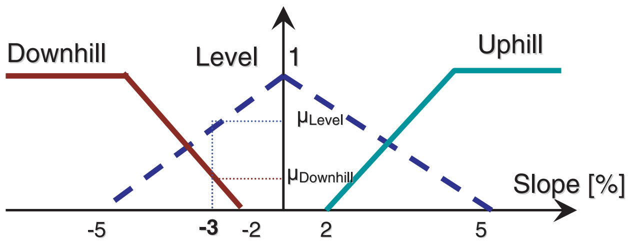

Figure 4: An example of the slope membership functions.

ship function is a design problem for fuzzy systems, although the properties of the system do not change significantly if the membership functions are modified slightly. For example, Figure 4 demonstrates a typical triangular membership function built for expressing the slope. The slope in the fuzzy language can be expressed by at least three membership functions defining particular slope ranges: “Downhill”, “Level” and “Uphill”. A point on this scale may have three values, one for each of the three membership functions.

The fuzzy rule base provides a formal methodology for indicating linguistic rules developed through a domain expert (for better understanding we are using a simple if-then form rather than a specific programming language – as mentioned above: LHS = *left-hand side*, RHS = *right-hand side*):

$$\text{Rule}(i):$$
$$\underbrace{\text{If } x_{i1} \text{ is } A_{i1} \text{ AND } x_{i2} \text{ is } A_{i2}, \ldots, \text{ AND } x_{im} \text{ is } A_{im}}_{\text{LHS}}$$
$$\text{THEN } \underbrace{y \text{ is } B_i}_{\text{RHS}} \tag{4}$$

where $i = 1, \ldots, n$, and n is the number of rules in a given fuzzy rule base; $j = 1, \ldots, m$, and m is the number of antecedents; x_{ij} are the input variables represented as premise variables; A_{ij} are the input fuzzy sets; and B_i is the output fuzzy set (consequences). As mentioned before, the rules are the core of a (fuzzy) knowledge-based system, reflecting the expert’s experience in interpreting the input parameters and determining the output quantity (domain-specific knowledge). The rules below, based on our example, relate two input variables and one output variable of our fuzzy system:

$$\begin{aligned} &\text{Rule}(1)\text{: If } x_1 \text{ is downhill AND } x_2 \text{ is Level} \\ &\qquad \text{THEN } y \text{ is Level} \\ &\text{Rule}(2)\text{: If } x_1 \text{ is Level AND } x_2 \text{ is uphill} \\ &\qquad \text{THEN } y \text{ is uphill} \end{aligned} \tag{5}$$

Note that different experts would perhaps produce a different collection of membership functions; however, due to the tolerance of the fuzzy systems to the level of approximation, the systems would eventually yield similar results, if all the experts have captured the main points of interest (Kosko 1991).

The inference engine is the kernel of a (fuzzy) knowledge-based system, which carries out the approximate reasoning task. Each rule for $i = 1, \ldots, n$, in equation 5 corresponds to a fuzzy rule. For each of the premises, an area corresponding to the membership function of the input variable is created.

The final result of fuzzy inference is a fuzzy set, which is obtained by aggregating the results of the individual rules. The result of the fuzzy inference is a fuzzy set. Defuzzification operates on the implied fuzzy sets produced by the inference engine, and combines their effects to yield a “nonfuzzy” decision output. The most commonly used method of defuzzification is the Center of Area (COA), which generates the center of gravity of the area below the membership function, and projects it onto the domain axis. The corresponding domain point is used as a numerical output value.

2.2. *Artificial neural networks*

An Artificial Neural Network (ANN), also referred to as Neural Network, is an information processing system that is inspired by the structure and functional aspects of biological neural networks of the human brain (Negnevitsky 2005). ANN represents a mathematical or computational model that consists of an interconnected group of artificial neurons, and processes information using a connective approach for computation. It is characterized by its ability to approximate an unknown input-output mapping through a training phase (Kosko 1991). Given a set of example pairs (input/output), a neural network can be trained to approximate a smooth function relating the data without requiring any initial dynamic or noise models. This ability comes from the distributivity of computations within the ANN and allows powerful learning capabilities (Lin and Lee 1996, Russell and Norvig 2003).

In most cases, an ANN is an adaptive system capable of changing its structure as a result of external or internal information that flows through the network during the learning phase. Neural networks are considered non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in the data. An example of ANN is shown in Figure 5.

As can be seen in Figure 5, ANNs have a large number of highly interconnected processing elements (neurons) that usually operate in parallel and are configured in a regular architecture. Each neuron in a neural network performs a predefined mathematical function $f(\ )$ to determine the strength of the firing from the neuron. The input neurons are connected to many other neurons in the next layer with various weights, resulting in a sequence of outputs, one per neuron. The layer that receives inputs $x_1, \ldots, x_m$ is called input layer, and typically performs no computation. The output of the network is called output layer, which may have single or multiple outputs, $y_1, \ldots, y_p$. Any layer between the input and the output layers is called a hidden layer. Each connection also has an associated weight, which determines the effect of the incoming input on the firing level of the neuron.

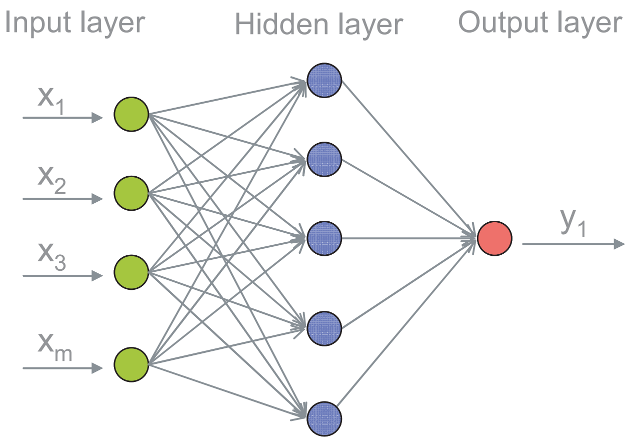

Figure 5: A neural network: an interconnected group of neurons in different layers.

As each neuron output is connected through the weights to other neurons in other layers (or the same), different types of neural networks can be specified based on the structure that organizes these neurons and the connection geometry among them. The feedforward neural network, as shown in Figure 5, is one of the first and arguably simplest type of neural network, designed by interconnecting several layers. It is feedforward, because no neuron output is an input to a neuron in the same layer or preceding layer, and the process moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) towards the output nodes. The other commonly used network architectures include the radial basis network, the Kohonen self-organizing map, and various recurrent neural networks (Kosko 1991).

The connection weights store the information, and their values are determined by a neural network learning process. It is through adjustment of the connection weights that the neural network is able to learn (Lin and Lee 1996). However, training an ANN is, in general, a challenging nonlinear optimization problem. The backpropagation algorithm is the most widely used algorithm to train a network (Zimmermann 1985). This algorithm is a gradient descent method of training – it uses gradient information to modify the network weights to decrease the error difference between the desired output and the network output, while the errors propagate backwards from the output neurons to the input neurons. However, the off-line backpropagation training is generally achieved through a substantial number of iterations to reach convergence. For real-time applications, using a Kalman filter for training is likely to speed up the process as an online algorithm (Williams 1992, Iiguni et al. 1992).

Training the network may be time consuming. It usually learns trough several epochs, depending on how large the network is. Basically, the network is trained for several epochs and the training is terminated after reaching the maximum number of epochs, or meeting a pre-defined thresholds. For the same reason, minimum error tolerance is used, provided that the difference between the network output and the known outcome is less than the specified value.

The critical issue in developing an ANN is this generalization: how well will the network approximate a function that is not in the training set? ANN, like other flexible nonlinear estimation methods can suffer from either underfitting or overfitting. During training, the network might learn too much. This problem is referred to as overfitting. Overfitting is a critical problem in most standard ANN architectures. Overfitting is especially dangerous because it can easily lead to predictions that are far beyond the range of the training data with many of the common types of ANNs. One of the solutions is early stopping, but this approach needs more critical attention as this problem is harder than expected. A network can also fail in fully approximating a function in a complicated data set, leading to underfitting. The underfitting can also happen when the input data are so similar that they appear as clusters (Negnevitsky 2005).

2.3. Evolutionary strategies

An evolutionary strategy (ES) is a local stochastic search strategy, which randomly modifies the nominal values of a present set of non-optimal object variables in order to find a setting that satisfies any given problem-specific conditions with a chosen accuracy (Rechenberg 1994). The changing is not arbitrary; it is done in a well-defined manner. The ES first uses an initial (parental) setting of values to create a certain number of different (offspring) versions out of it. Only in the very beginning, any modification, i.e., mutation is completely randomly chosen. The initial mutability rate is a strategy parameter and also subject to change. Other important strategy parameters are, for instance, the number of individuals (parents and offspring). Within a so-called Meta-ES, not only the object parameters but any of the strategy parameters can also be optimized.

All different versions of the set of nominal values for the object variables will be evaluated by testing them against a given (problem-dependent) quality function. The latter combines all applicable (known) condition equations. By comparing the different sets, the ES sorts out the most unfit versions, i.e., individuals. The remaining individuals create a new generation set of object variables. Memorizing successful mutations of past generations enables the ES to define new and promising mutability rates for the next generation. Modern ESs allow for an efficient adaptation of these rates (in the form of a covariance matrix). Leaving a margin for randomness is necessary to secure that the ES algorithm is able to escape from insufficient local optima.

An ES at least includes mutation and selection. More elaborate versions make use of some additional evolutionary mechanisms, e.g., recombination. In Alvers (1998) the reader can find illustrations of the basic concepts for several optimization methods.

ES can be used to implement technical optimization algorithms for real-world problems. They are universal, undemanding, close to reality, easy to implement, and can be considered as a compromise between volume and path orientated searches for the optimal solution. Yet, there is no guarantee to find the global optimum, and the convergence speed might be less compared to methods that are tuned to a specific problem.

In general, optimization strategies can be classified as follows (Rechenberg 1994):

- global deterministic search strategies (e.g., systematic scanning),
- local deterministic search strategies (e.g., gradient strategies),
- global stochastic search strategies (e.g., Monte Carlo method),
- local stochastic search strategies (e.g., evolutionary strategies).

In Geodesy, the traditional least-squares approach as an example for a gradient strategy (GS) is commonly used. GSs are characterized by taking deterministic steps that needs a kind of a pre-testing to gain information on the "best" search direction. This collection of information comes without any improvements yet, and it may take much time. Only at the actual job steps (hill climbing towards the optimum) is there a gain in quality. On the other hand, it is the maximal possible improvement. In opposition to that procedure, ESs just take spontaneous small random steps (uphill diffusion). The chance for a big win per single random step is quite low. But (at least in the linear case) there is (on average) a 50% chance to gain a small win per single step (see Rechenberg 1994).

In practice, the ES-algorithm will start with an arbitrary initial guess (0th generation), evaluate it using the objective function, sort the offspring, and select the best-fitted individuals. Only these remaining individuals will be mutated (in a more or less reasoned and adapted but still randomly way), and define the next generation. These fundamental steps are repeated loop-wise until a certain given threshold value for the quality of the solution is reached. The only real condition for an ES to function is that the given system/problem does not show a chaotic behaviour. The algorithm actually just relies on the existence of strong causality (similar causes lead to similar effects)[2].

Nowadays, there are many different versions of ES algorithms available. Basically, they differ by the handling of the strategy parameters, e.g., number of parents and successors, selection with or without parents included, mutation rate adaptation, usage of historic/past information on successful guesses, parallel populations, mutation of the strategy parameters itself (Meta-ES), etc. The choice of a specific setting may depend on the practical needs. But one major advantage of an ES is that usually the user does not need to have deep insight into the given problem. Remember that nature uses evolutionary strategies to optimize real-world problems.

In order to rate the applicability of any given optimization strategy (stochastic or deterministic, globally or locally), realistic (non-trivial) objective test functions (i.e., quality functions) should be used, having at least the following properties: non-linear, non-separable, free-scalable, not solvable by simple "hill-climbers," real-valued parameters, strong causality, and a single global extremum/optimum only.

ESs make use of biological methods of optimization. Several mechanisms are being deployed, mainly:

- mutation (of individual genes, chromosomes, entire genome),
- recombination (replacement or mixture, intermediary or discrete),
- selection (ensures the necessary limitation of the number of individuals),
- isolation (in space and/or time, weak or strong).

In nature, quality evaluations (fitness tests) occur with respect to the speed of adaptation to a changing environment – the faster the better. Time is measured in numbers of generations needed to adapt. For technical implementations the combination of time and (possibly accruing construction) costs is much more important. In purely mathematical experiments on a computer this aspect becomes negligible. Applying an ES does not mean to simply copy natural processes – it is more important to understand the principles and functionality behind it and to adapt it to a technical/artificial environment. Despite all analogies to biology, there may be significant differences. Bear in mind, that nature does *not* always

- keep the population size constant (but most ES algorithms will do so),
- select the most fitted individuals (but probably all ES algorithms will do so),
- allow for multi-recombination (more than two parents per child may lead to a more robust or fitter offspring; in fact you can also find multi-recombination in nature, e.g., viruses and bacteria which are very successful creatures indeed).

Within the last decades EAs have seen many improvements, starting from very simple algorithms. Almost at the same time (late 1960s, early 1970s) Genetic Algorithms and Evolutionary Strategies became a research topic. Traditionally, GAs are more common in the US (e.g., Holland (1975), Koza (1992), Goldberg (1993)) whereas ESs were profoundly studied for instance in Germany (e.g., Rechenberg (1994), Hansen und Ostermeier (1996 and 2001), Ostermeier (1997)). Because of a widespread confusion between GA and ES, the main properties/differences of these two EA branches will be outlined in Table 1.

3. Applications of AI techniques in geodesy – examples

In the following we will present three examples of applications of AI techniques in Geodesy:

- knowledge-based deformation interpretation,
- intelligent personal navigation,
- gravity field determination using evolutionary strategies.

3.1. Knowledge-based deformation interpretation

In recent years damage caused by rockfalls and landslides has been increasing, as well as the number of killed or injured persons, due to a spread of settlements in mountain areas. In addition, many global climate change scenarios predict an increase in the probability of heavy rain, which is a primary trigger

2 "Weak causality" means that equal causes lead to equal effects, which is not applicable to reality and therefore is called a "weak" principle.

Table 1: Genetic Algorithms vs. Evolutionary Strategies.

Genetic Algorithms (GA)	Evolutionary Strategies (ES)
imitation of the cause	imitation of the effect
mutation at the genotype	mutation at the phenotype
more experimental, less theoretical	sustained theoretically
mainly driven by recombination & selection (low mutation rate, soft selection)	mainly driven by mutation & selection (high mutation rate, rough selection)
very large population size	relatively small population size
no waste of any offspring	variable selection pressure is possible
parents of good quality will be reproduced more likely (proportional to its fitness)	all parents have the same chance to become reproduced (same probability)
faster convergence in the beginning	needs some time for step size adaptation
poor fine tuning capabilities	parameters can be adjusted very precisely
strong causality may become violated	the more robust the slower
recombination result mainly just reflects the way of encoding	there is no need to transform into other representations (e.g., binary)

for rockfalls and landslides. This causes an urgent need for highly effective and reliable tools for monitoring rockfalls and landslides at an operational level.

The rising importance of rockfall and landslide monitoring is clearly also reflected by a huge number of research projects. For example in its last two framework programs, the European Commission has positioned research about "Natural Hazards" and "Disaster-Management" as a priority topic.

The core of geo-risk management consists of identifying, understanding and mitigating risk by reducing the probability or consequences of rockfalls. In the literature, several geo-risk management and geo-monitoring systems can be found – most notable are Fujisawa (2000), Kahmen and Niemeier (2003), McHugh (2004), Scaioni et al. (2004), and Scheikl et al. (2000). Examples of systems used in practice are GOCA (2008), and GeoMoS (2008). The main application field of these tools is monitoring and analyzing deformations – however, they offer no possibility for deformation interpretation. Currently, this is done by human experts. Experts from geology and civil engineering interpret deformations on the basis of a large number of data records, of documents and of knowledge of different origins. In this context SLIDISP (Liener et al. 1996) should be mentioned, which represents an automated system for building risk potential maps. The interpretation of the corresponding map is still a manual process involving expert knowledge.

An evaluation of the risk of a rockfall or landslide is well suited for a KBS. First, the problem is well defined – the solution requires a heuristic approach and expert knowledge. Second, knowledge and data from different fields and sources have to be considered. Third, experts in this area are not widely available. Fourth, geo-risk objects that need monitoring and need a deep interpretation including a huge number of data sets are increasing. Consequently, a knowledge-based support system for risk classification is a valuable tool – it supports the experts in their task and provides a first risk assessment.

The interdisciplinary research project "*i-MeaS*" ("*An Intelligent Image-Based Measurement System for Geo-Hazard Monitoring*") has been launched with the purpose of supporting the expert in her interpretation process for geo-risk objects (i-MeaS 2010). The system produces on-line information about ongoing deformations and supports issuing alerts in case the deformation behavior exceeds a predefined limit.

The system uses different data and knowledge:

- *measurement data*, captured by different sensor systems, e.g., total stations, geotechnical sensors, etc.
- *local weather data*, like local temperature, the amount of precipitation, the kind of precipitation, etc.
- *global meteo data*, which are provided by meteorological service like the Austrian Meteorological Service or the Swiss Meteorological Service
- *expert knowledge* (i.e., domain knowledge), which has been collected in an intensive knowledge acquisition process.

The approach discussed here is based on a data-driven forward-chaining rule-based system, which is capable of handling all the mentioned sensor types and data. Such a system is sensitive to the input of new data and infers conclusions from them. They are therefore well suited in application domains where new data (e.g., new measurement values) arrive frequently and have to be integrated in the reasoning process.

We already discussed the basic components of a rule-based system, and we showed some rules in "pseudo-code", but we have not presented a more detailed view of rules so far. We said that a rule consists of two parts, the left-hand side and the right-hand side. If the expert has formulated a rule of the form "*If we have had heavy rain over the last three days and the soil was very wet before, then* ...", the question arises what is "heavy rain" and when is the soil "very wet." Usually, we have measurement data that reflect a specific amount of rain (e.g., 20 mm per hour), but the system is unable to conclude whether this is heavy rain or not. As mentioned in Section 2.2, the numerical values can be translated into linguistic concepts by fuzzification. The use of such an abstraction procedure permits us to write rules in terms of easily-understood word descriptors, rather than in terms of numerical values.

As an example, we want to define a rule which calculates a risk factor. In the LHS of the rule "risk_precipitation" below, we check whether there is an element of type *meteo_local* in the working memory where its slot *precipitation* is "high" or "very_high" and whether there is an element of type *meteo_global* where the same slot value is "high" or "very_high". The RHS consists of the instruction to include a new element of type risk_factor into the working memory. This new fact comprises a mathematical formula where the risk factor is increased.

```
(defrule risk_precipitation
   (declare (salience 200))
   (meteo_local {precipitation == high
   || precipitation == very_high})
   (meteo_global {precipitation == high              (6)
   || precipitation == very_high})
=>
   assert (risk_factor (risk + 1)))
)
```

The *salience* rule property allows the knowledge engineer to assign a priority to a rule. If more than one rule has a satisfied left-hand side, then a rule instance with the highest priority will be fired. The selection is made by conflict resolution already discussed above.

Let us describe the architecture of the proposed knowledge-based interpretation system. It is organized as a two-level approach:

Level 1: The detection of an actual risk factor of the landslide. On the basis of a questioner answered by experts inspecting the site, the actual state of the landslide is analyzed. This analysis is based on questions regarding the geological and environmental situation of the landslide. The expert has the possibility to answer the questions on-site by means of a mobile device (netbook, table pc, etc.). The result of this first step is a grading of the landslide into one of five predefined independent risk levels.

Level 2: The processing of meteorological, geological and geodetic knowledge and data. On the basis of the determined initial risk factor, the continuously captured measurement data, such as deformation vectors and/or meteorological data are employed to predict the behavior of the landslide. In a final step, this information is condensed into a current risk factor, which may be changed if the environmental parameters change. A simplified structure of the architecture of the system is shown in Figure 6.

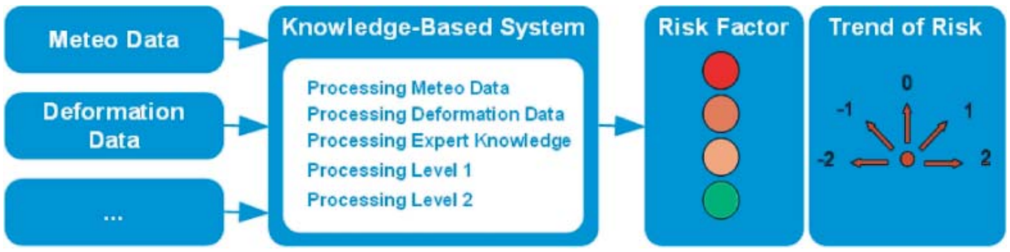

Figure 6: Simplified architecture of the intelligent system for geo-risk assessment.

Currently the whole system is in prototype state. The knowledge base has been divided into separated units for the processing of meteorological data, deformation data and geological data. Furthermore an explanation tool, which helps the user to understand the decision making, is under development. For the implementation of the rule-based system, JESS has been used. The single processing units have been embedded into a JAVA framework, which itself has been integrated into a global MATLAB program. The system for processing *Level 1* has been implemented and tested. A deep evaluation (system performance has been compared with several experts from geology) shows promising results – the system behaves like a human expert. The Level 2 system is currently under development – knowledge acquisition has been finished and the RBS is implemented. Evaluation of this part has been started. The implementation process will be completed by a general evaluation process of the whole processing sequence (Level 1 and Level 2).

3.2. *Intelligent personal navigation*

The increasing market of mobile information systems for pedestrians (e.g., information systems for city tours or public buildings) requires the precise and reliable provision of the current position. In contrast to the well-established vehicle navigation systems that use the combination of position sequences with digital maps (e.g., map-matching techniques) pedestrians frequently move outside from digitally acquired roads/trajectories (Thienelt et al. 2008).

Due to the complexity of sensory data entities and their dynamic characteristics, as well as their rapidly changing availability in varying environments, an "intelligent" navigation approach is preferred instead of a classical dead reckoning (DR) algorithm (Moafipoor et al. 2007). For our example, "intelligent" stands for using ANN and fuzzy logic to estimate DR components of the overall dynamic and parametric models.

An effective representation of knowledge is one of the key issues for most AI-related solutions. In the example presented here, the data/knowledge acquired from the sensors represents body locomotion in the form of descriptive definitions and classes of objects, as well as their interpretations. Table 2 lists all of the measurements, as delivered by the sensors and used in the current intelligent personal navigation (PN) prototype, which can provide input parameters to parameterize body locomotion, step length/direction/altitude modeling functions.

Table 2: PN sensors and their measurements contributing to the parameterization of the locomotion model.

Sensor	Sensor Measurements
Accelerometer	Step events $\lvert a\rvert_{xyz}$, $\lvert a\rvert_{xy}$, $\lvert a\rvert_z$ Std($\lvert a\rvert_{xyz}$), Std($\lvert a\rvert_{xy}$), Std($\lvert a\rvert_z$) Max($\lvert a\rvert_{xyz}$), Min($\lvert a\rvert_{xyz}$) Tilt (roll and pitch angles at rest)
Gyroscope	Angular rate
Magnetometer	Roll, pitch, heading
Barometer	Altitude Std(Δh) – $\sum(\lvert\Delta h\rvert)$
Step sensors	– Step events
External data	– Operator's height, age, weight

In Table 2, $|a|_{xyz}$, $|a|_{xy}$, and $|a|_z$ are the magnitudes of the acceleration vector during a single step in 3D, horizontal, and vertical directions, respectively; Std($|a|_{xyz}$), Std($|a|_{xy}$), and Std($|a|_z$) are the corresponding standard deviations of the acceleration vector norms; Max($|a|_{xyz}$) and Min($|a|_{xyz}$) are the maximum and minimum values of the acceleration magnitude for each step – note that the tilt, in the form of roll and pitch angles, is available only when the sensor is at rest (Jekeli 2001). The IMU measurements are used to determine the category of the locomotion pattern. The barometer provides information about body locomotion, such as change of altitude, Δh, and its standard deviation, Std(Δh), and the total displacement in altitude, $\sum(|\Delta h|)$. Step sensors are used to detect step events and to derive the step length (SL). Lee and Mase (2001a) showed that the standard deviation, the maximum, minimum, and the SL parameters are related to the speed of walking, whereas the mean values and height variation are related to the terrain gradient.

The first approach to approximate SL modeling was a Radial Basis Function (RBF) Neural Network (Grejner-Brzezinska et al. 2007). The RBF network was designed to perform input-output mapping based on the concept of a locally tuned and overlapping receptive field basis function structure. The RBF network was trained by the hybrid learning rule, unsupervised learning in the input layer (i.e., the sensor measurements in Table 2) and supervised learning in the output layer (i.e., SL is calibrated during GPS availability). Experiments showed an accurate ANN SL modeling, at a 0 ± 5 cm level. However, in this approach, it was not possible to extract structural knowledge (rules) from the training data, nor could we integrate special information about the environment (e.g., staircase constraints, map/image data) into the RBF neural network structure in order to tune the learning procedures. To overcome these limitations, a fuzzy logic system for SL modeling has been proposed and implemented (Moafipoor et al. 2008a). In order to determine the final design and implementation in the intelligent PN, both approaches have been studied and compared by Moafipoor et al. (2008b).

With the heading measured by the gyro and/or magnetometer, and with the pre-calibrated SL predicted by the ANN modeling, the DR of the next position can be determined. However, this kind of point-to-point navigation trajectory reconstruction has significant shortcomings, particularly in its inability to accommodate outlier detection, since no continuity/predictive model for the trajectory is formed. Outliers can be caused, e.g., by imprecise SL prediction and uncalibrated magnetometer/gyro step direction (SD) determination, which can easily bias the determination of subsequent positions. An approach that can address these shortcomings is the consideration of a human body model. Hausdorff et al. (2001) showed that human motion demonstrates specific characteristics, which make the definition and identification of generic models of human motion feasible. To this end, it may be possible to customize a prediction model for SL and SD values based on our knowledge of a human locomotion model. This framework was represented in the form of a Kalman filter, called DR-KF for basic human activities, including stumbling, walking, running and climbing stairs. Equation 7 shows the general structure of the DR-KF, where the prediction model is established based on the body locomotion pattern:

If the locomotion pattern (L_P) is [stumble/stand/walk/run/climb] then

$$\begin{cases} X_k = \Phi_k^{L\_P} X_{k-1} + u_k^{L\_P} \\ Z_k = H_k X_k + v_k \end{cases} \tag{7}$$

where X represents the state vector, including position, SL, SD, and step altitude (SA); Φ^{L_P} is the transition matrix of the prediction model; u^{L_P} is the random error vector of the prediction model; Z is the observation vector; H is the observation matrix; and v expresses the random observation error (Moafipoor et al. 2008b). This kind of fuzzy control model in the form of a Kalman filter is known as Takagi-Sugeno (T-S) fuzzy models. The T–S fuzzy model is based on the observation that a modeling problem can be separated into local approximations. The local approximations are then smoothly interpolated to obtain the global model (Simon 2003).

Figure 7 shows the DR navigation trajectory, reconstructed using FL-based and ANN-based SL modeling with integrated gyro and magnetometer sensor data. The numerical results were conducted on a series of datasets – for this test, the operator walked one and a half indoor loops for about 97 m in 2 minutes.

The square symbols in Figure 7 represent the ground control points that were followed by the operator, representing the reference trajectory. As shown in the figure, several SD biases were identified and, subsequently, fixed. The locus of SD outliers was identified mostly around the corners, where a large difference between the prediction and observation models was estimated. In contrast, due to correct SL modeling based on accurate locomotion pattern estimation, there were few outliers marked for SL value. The statistical results of the reconstructed trajectory using different SL observations are shown in Table 3, indicating a similar performance for both SL modeling methods.

3.3. *Gravity field determination using ES*

The main goal of this example is to prove the applicability of an evolutionary strategy to a typical over-determined and inverse problem in Satellite Geodesy. Adjustment techniques like the traditional least-squares approach based upon the gradient search require the inversion of a normal equation matrix. Depending of the resolution of the gravity field model and possibly other unknown force field parameters, this matrix can become very large in size. This may impose a heavy computational burden. Besides that, a gradient search in general implies some mathematical preconditions (e.g., derivability) on the functional model.

On the contrary, ES algorithms do not require derivability nor inversion of any matrix. In fact, we do not have to care about all the theoretical as well as practical problems associated with the inversion of huge normal equation matrices (e.g., ill-conditioned matrix, numerical instability, etc.). The computational burden in this case lies solely in the multiple and therefore time-consuming evaluation of a quality function. Due to the expected growing speed of

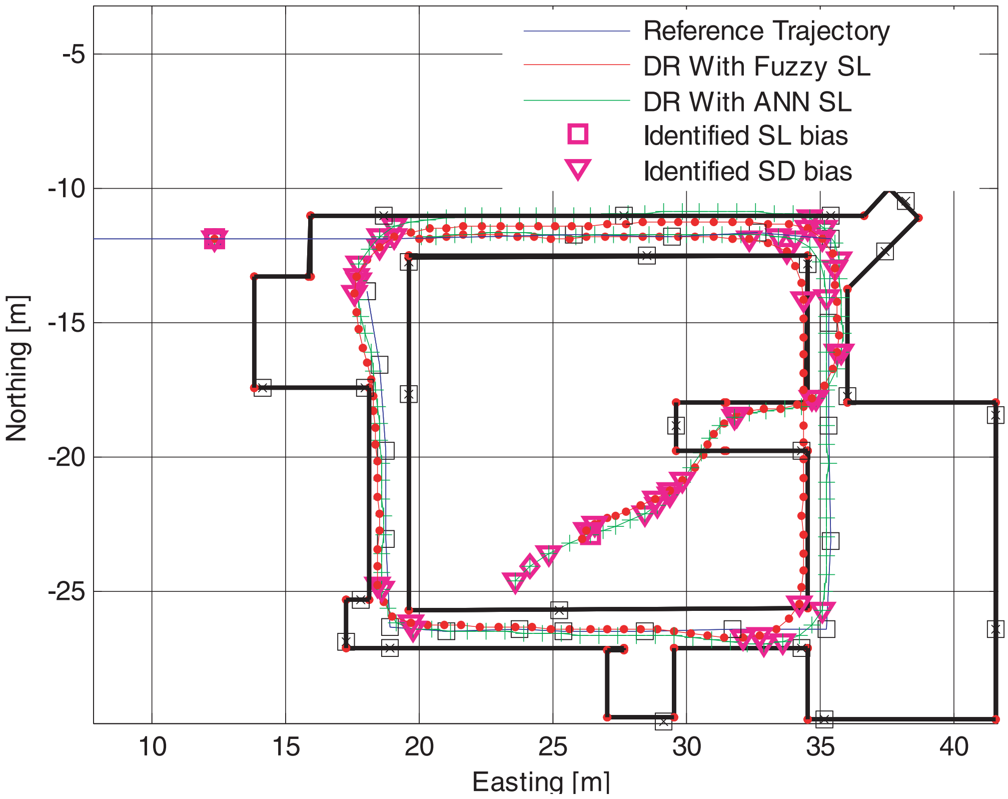

Figure 7: Test area floor plan and DR-KF trajectory reconstruction based on FL and ANN SL modeling integrated with gyro/magnetometer heading, adjusted by the DR-KF module.

Table 3: Statistical fit to reference trajectory of DR trajectories generated using SL predicted with FL and ANN, and the integration of gyro/magnetometer compass heading adjusted by the DR-KF module.

97 m; one and a half indoor loops		Mean [m]	Std [m]	Max [m]	End Misclosure [m]	CEP (50%) [m]
SD modeling	SL modeling					
Gyro/magnetometer heading	FL	0.90	1.34	1.2	1.2	0.39
	ANN	0.88	1.25	1.1	1.3	0.41

our computers, this limitation will be overcome in the future even for the determination of high-resolution gravity fields. In this case, the original inverse problem could always be solved in a direct manner. This application shall just prove the general concept.

The task of gravity field determination requires the finding of an optimal set of spherical harmonics c_{nm} and s_{nm}, representing Earth's gravity field. As an example we choose a simple 4×4-gravity field model ($n_{max} = m_{max} = 4$). Under certain assumptions (Mai 2005) this leads in total to 21 unknowns (object parameters) – therefore we are dealing with a 21-dimensional optimization problem.

Earth's gravity field directly influences the motion of an orbiting satellite. In order to determine our unknowns, a certain number of (observed or simulated) satellite positions $\mathbf{r}_i^s$ are given. In this case, we get a boundary value problem in Satellite Geodesy.

Any change (due to the optimization process) in the nominal values for the spherical harmonics will lead to a corresponding change in calculated satellite positions $\mathbf{r}_i^c$. Comparing these position vectors with the simulated ones yields a number of deviations/residuals $\Delta\mathbf{r}_i := \mathbf{r}_i^s - \mathbf{r}_i^c$. These differences should not exceed a chosen threshold based on the accuracy of orbital observations. Depending on the norm, the objective function (quality criterion) may be defined as

$$Q = \sum_{i=1}^{N} \|\Delta \mathbf{r}_i\| \rightarrow \min \tag{8}$$

where N is the total number of given/available satellite positions (the following example uses $N = 90$ vectors). The simulation was done by applying numerical orbit integration taking the well-established UTOPIA software package, as provided by the University of Texas at Austin (it is based on a Krogh-Shampine-Gordon numerical integrator). The following Keplerian elements were used as initial values:

$$a_0 = 7000 \text{ km}, \quad e_0 = 0.007, \quad i_0 = 70^\circ,$$
$$\Omega_0 = 0^\circ, \quad \omega_0 = 0^\circ, \quad M_0 = -70^\circ.$$

Applying a fixed integration step size (= output step size) of 60 seconds leads to a simulated orbital arc of 90 minutes length. At an intentionally low orbital height of approximately 630 kilometers (to be fairly sensitive to gravitational effects), the satellite will almost complete one single revolution within that time interval.

Regarding the starting of the ES algorithm, it is not necessary to thoroughly consider a fitting initial guess about the unknowns. We might even just take zeros. For our example, a (1,40)-ES was realized. For each new generation there is only one parent but 40 descendants, and *only* the (mutated) offspring are subject to selection afterwards. This is denoted by the comma within the round bracket, following the usual ES-notation – whereas a plus sign would imply that *both* parents and offspring are taking part in the selection step. The optimization procedure contains adaption phase(s) of the covariance matrix (covariance matrix adaptation – CMA), which itself describes the mutability of the unknowns.

Within this example only a simple mutation of the genes, i.e., random change (within an adapted interval) of the nominal values of the individual spherical harmonics was applied. Other possible types of mutation, e.g., of the chromosomes (Rechenberg 1994), are not implemented yet.

Depending on the chosen threshold value (= termination quality Q^*), the optimization runtime can vary greatly. In general, most of the time is spent on the adaption. With λ denoting the number of offspring, and G being the number of generations it takes to reach the threshold value, the computational effort can be expressed by the total number of function calls (objective function evaluations) $\lambda \cdot G$.

Figure 8 shows all of the interim results of a single ES optimization run. Adaption phases of the covariance matrix are clearly visible. In practice processing time can be saved by incorporating any available problem-specific pre-knowledge. The left plot illustrates the residuals of the unknowns which is only available in a simulation, where the optimal solution is actually known).

The final (optimized) values for all unknowns were compared to the original spherical harmonics (based on the Joint Gravity Model JGM-3) that were used for the satellite orbit simulation using UTOPIA. The termination quality was set to $Q^* = 1/1000$ mm which is intentionally way beyond the accuracy

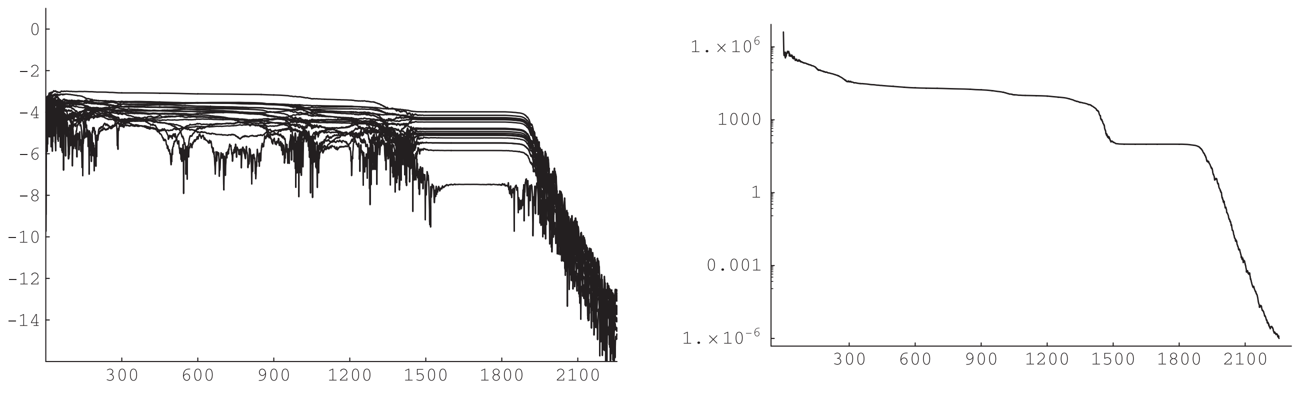

Figure 8: Logarithm of the absolute residual values of the unknowns vs. generation (left), and quality following Equation 8 in m vs. generation (right).

levels of any of today's available satellite observation techniques (GPS, SLR, etc.).

As a result, the final values are, on average, precise to the level of 10^{-13} that corresponds to a number of 6 to 7 significant digits that could actually be reproduced. This gives an idea about the achievable absolute resolution for similar gravity field determinations. The applied algorithm did not make use of any pre-knowledge from celestial mechanics, or physics. Reasoned conditions might be imposed to the solution of this inverse problem. As an example, there exist certain integrals of motion that could be accounted for by adding penalty terms (Alvers 1998) to the quality function. The algorithm can sort out unfitted solutions more quickly, but only if condition evaluations take just a little extra time.[3]

4. Conclusion and outlook

In this paper we have presented an overview of AI techniques in Geodesy. Three different techniques, namely knowledge-based systems (including fuzzy logic), artificial neural networks and evolutionary strategies have been addressed. All these methods have been illustrated by some practically useful examples.

All the described techniques have their advantages and disadvantages. Knowledge-based systems require domain knowledge mainly acquired from experts. This knowledge acquisition is the most important and costly part in the development of such a system. Negnevitsky (2005) presents an overview of knowledge engineering and the development process (including development tools) in more details. Due to the explicit (symbolic) representation of knowledge, the user is able to query the system for an explanation of how a result has been obtained, which knowledge facets are important for the reasoning process, etc. Such an explanation capability is indispensable for an acceptance of the system's result by the user.

For the other two techniques described in this paper, an explicit representation of domain knowledge is not required, which makes the development easier. For instance, in Artificial Neural Network, an approximation of the input-output relation is set in a training phase by well-chosen examples. The weight of the connections between computing elements are fined-tuned by propagation algorithms. The knowledge is not explicitly (symbolically) represented, but is distributed over the weights of the connections. This sub-symbolic representation makes the generation of explanation of the solution or the solution finding process impossible.

We expect an increasing use of AI techniques in Geodesy, mainly because tools become more and more available. The integration of AI techniques into the development of Geodetic applications greatly simplifies system creation by allowing the programmer to deal with more abstract concepts like rules instead of low-level constructs like if-statements in C-programs or compare-and-branch sequences in assembler programming.

Acknowledgements

The research presented in this paper has been supported by IAG Sub-Commission 4.2 – Working Group 4.2.3 ("Application of Artificial Intelligence in Engineering Geodesy"), the Austrian Science Foundation (FWF) Project L514 *i-MeaS* ("An Intelligent Image-Based Measurement System for Geo-Hazard Monitoring") and by the US National Geospatial-Intelligence Agency Project "Seamless and Reliable Personal Navigator".

References

Alvers, M., Zur Anwendung von Optimierungsstrategien auf Potentialfeldmodelle, Dissertation, Fachbereich Geowissenschaften, Freie Universität Berlin, 1998.

Brezing, A., Entwicklung eines Expertensystems zur wissensbasierten Deformationsanalyse, PhD Thesis, RWTH Aachen, 2000.

Chmelina, K., Wissensbasierte Analyse von Verschiebungsdaten im Tunnelbau, PhD Thesis, Vienna University of Technology, 2002.

Cooper, G.-F., The computational complexity of probabilistic inference using Bayesian Belief Networks, Artificial Intelligence (42)2–3 (1990), 393–405.

Debenham, J., The implementation of expert, knowledge-based systems, Proceedings of the 11th International Joint Conference on Artificial Intelligence 1 (1989), 221–226 .

Elster, K.-H., Modern mathematical methods of optimization, Wiley-VCH, 1993.

Ferreira, J.-F., Pinho, C. and Dias, J., Active exploration using Bayesian Models for multimodal perception, Lecture Notes in Computer Science 5112 (2008), 369–378.

Fujisawa, K., Monitoring technique for rock fall monitoring, Internal Report, 2000.

GeoMoS, http://www.leica-geosystems.com (accessed September 2010).

3 Remark: Please be aware that the whole optimization was done without any inversion of a (normal equation) matrix. For high-resolution gravity field models there are hundred thousands of unknowns, and the commonly applied least-squares methods have to invert a correspondingly huge normal equation matrix. This imposes a heavy computational burden and it takes much time and space in terms of memory.

Ghahramani, Z. and Roweis, S.-T., Learning nonlinear dynamical systems using an EM algorithm, Advances in Neural. Information Processing Systems, Vol. 11, MIT Press, 1999.

GOCA, http://www.goca.info (accessed September 2010).

Goldberg, D. E., Genetic algorithms in search, Optimization and Machine Learning, Addison-Wesley, Reading, Massachusetts, 1993.

Grejner-Brzezinska, D. A., Toth, C. K. and Moafipoor, S., Pedestrian tracking and navigation using adaptive knowledge system based on neural networks and fuzzy logic, Journal of Applied Geodesy 1(3) (2007), 111–123.

Haberler-Weber, M., Huber, M., Wunderlich, T. and Kandler, C., Fuzzy system based analysis of deformation monitoring data at eiblschrofen rockfall area. Journal of Applied Geodesy 1 (2007), 17–26.

Hausdorff, J. M., Ashkenazy, Y., Peng, C. K., Ivanov, P. C., Stanley, H. E. and Goldberger, A. L., When human walking becomes random walking: Fractal analysis and modeling of gait rhythm fluctuations, *PHYSICA A* 302 (2001), 138–147.

Hansen, N. and Ostermeier, A., Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaption, in: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), 1996, 312–317.

Hansen, N. and Ostermeier, A., Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9(2) (2001), 159–195.

Heckathorn, D.-D., Cognitive science, sociology, and the theoretic analysis of complex systems, *The Journal of Mathematical Sociology* (14)2 (1989), 97–110.

Heine, K., Beschreibung von Deformationsprozessen durch Volterra- und Fuzzy-Modelle sowie Neuronale Netze, Deutsche Geodätische Kommission, C, 516, München, 1999.

Heinert, M. and Niemeier, W., From fully automated observations to a neural network model inference: The bridge "Fallersleben Gate" in Brunswick, Germany, 1999–2006. J. Appl. Geodesy 1 (2007), 671–680.

Holland, J. H., Adaptation in natural and artificial systems, Ann Arbor, MI: University of Michigan Press, 1975.

Iiguni, Y., Sakai, H. and Tokumaru, H., A real-time learning algorithm for a multilayered neural network based on the extended kalman filter, IEEE Trans. Signal Process 40(4) (1992), 959–966.

i-MeaS, http://info.tuwien.ac.at/ingeo/research/imeas/index.html (accessed September 2010).

Jekeli, C., Inertial navigation systems with geodetic applications, Walter de Gruyter: Berlin, Germany, 2001.

Jess, http://www.jessrules.com (accessed September 2010).

Kahmen, H. and Niemeier, W., OASYS – integrated optimization of landslide alert systems, In: Österreichische Zeitschrift für Vermessung und Geoinformation 91 (2003), 99–103.

Kuhlmann, H., An expert system for bridge monitoring. 7th International FIG-Symposium on Deformation Measurements, Banff, Canada, 1993.

Kusiak, A., Computational intelligence in designed manufacturing, Wiley-Interscience, 2000.

Kutterer, H., On the role of artificial intelligence techniques in engineering geodesy, in: Proceedings of Second International Workshop (AIEG 2010) – Application of Artificial Intelligence and Innovations in Engineering Geodesy, Braunschweig (2010), 7–9.

Kosko, B., Neural networks and fuzzy systems, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1991.

Koza, J. R., Genetic programming, on the programming of computers by means of natural selection, sixth printing, A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England, 1992.

Lee, S. W. and Mase, K., Recognition of walking behaviors for pedestrian navigation, Proceedings of the 2001 IEEE Conference on Control Applications (CCA01), Mexico City, Mexico (2001), 1152–1155.

Liener, S., Liniger, M., Krummenacher, B. and Kienholz, H., Slidisp – a procedure to locate landslide prone areas, in: Seventh International Symposium on Landslides, ISL 1996.

Lin, C. T. and Lee, C. S., Neural fuzzy systems, Prentice Hall: Englewood Cliffs, NJ, 1996.

Mai, E., Spektrale Untersuchung GPS-ähnlicher Orbits unter Anwendung einer Analytischen Bahntheorie 2. Ordnung, PhD Thesis, Technische Universität Berlin, 2005.

Marr, D., Artificial intelligence – a personal view, Artificial Intelligence, Vol. (9)1, Elsevier (1977), 37–48.

Martin, S. and Kutterer, H., Modellierung von Bauwerksdeformationen mit Neuro-Fuzzy-Verfahren, In: Brunner, F. K. (eds.): Ingenieurvermessung 2007 – Beiträge zum 15. Internationalen Ingenieurvermessungskurs, Graz, Wichmann, Heidelberg (2007), 231–242.

McHugh, E., Video motion detection for real-time hazard warning in surface mines, In: NIOSHTIC No. 20026093, 2004.

Miima, J. B., Artificial neural networks and fuzzy logic techniques for the reconstruction of structural deformations, Geodätische Schriftenreihe, TU Braunschweig, Germany, 2002.

Moafipoor, S., Grejner-Brzezinska, D. A. and Toth, C. K. (2007). Adaptive calibration of a magnetometer compass for a personal navigation system. Proceedings of International Global Navigation Satellite Systems Society IGNSS Symposium, Sydney, Australia, CD-ROM.

Moafipoor, S., Grejner-Brzezinska, D. A. and Toth, C. K., A fuzzy dead reckoning algorithm for a personal navigator, NAVIGATION, 55(4) (2008a), 241–255.

Moafipoor, S., Grejner-Brzezinska, D. A. and Toth, C. K., Multi-sensor personal navigator supported by adaptive knowledge based system: Performance assessment, In: Proceeding of the IEEE/ION PLANS 2008 Meeting, Monterey, California, CD ROM, 2008b.

Negnevitsky, M., Artificial intelligence – a guide to intelligent systems, Addison Wesley, 2005.

Neuner, H. and Kutterer, H., Modellselektion in der ingenieurgeodätischen Deformationsanalyse, in: Wunderlich, T. (eds.): Ingenieurvermessung 2010, Wichmann, Berlin (2010), 199–210.

Ostermeier, A., Schrittweitenadaption in der Evolutionsstrategie mit einem entstochastisierten Ansatz, PhD Thesis, Technische Universität Berlin, 1997.

Rechenberg, I., Evolutionsstrategie '94, Frommann-Holzboog, Stuttgart, 1994.

Reiterer, A., Entwicklung und Erprobung eines schrittmotorgesteuerten Digitalnivelliers. Diploma Thesis, Vienna University of Technology, 2001.

Reiterer, A. and Egly, U., AIEG 2008 – proceedings of first international workshop – application of artificial intelligence and innovations in engineering geodesy, Vienna, Austria, 2008.

Reiterer, A., Egly, U., Heinert, M. and Riedel, B., AIEG 2010 – Proceedings of second international workshop – application of artificial intelligence and innovations in engineering geodesy, Braunschweig, Germany, 2010.

Rehr, I. and Kutterer, H., Effizienzoptimierung ingenieurgeodätischer Prozesse im Bauwesen. In: Schriftenreihe des DVW – Interdisziplinäre Messaufgaben im Bauwesen, Wißner Verlag (2010), 307–321.

Riedel, B. and Heinert, M., An adapted support vector machine for velocity field interpolation at Baota Landslide. In: Proceedings of First International Workshop (AIEG 2010) – Application of Artificial Intelligence in Engineering Geodesy, 2008.

Russell, S. and Norvig, P., Artificial intelligence – a modern approach. Prentice Hall, Pearson Education International, Upper Saddle River, New Jersey, 2003.

Scaioni, M., Giussani, A., Roncoroni, F., Sgrezaroli, M. and Vassena, G., Monitoring of geological sites by laser scanning techniques, IAPRSSIS 35 (2004), 708–713.

Scheikl, M., Poscher, G. and Grafinger, H., Application of the new automatic laser remote system (ALARM) for the continuous observation of the mass movement at the Eiblschrofen Rockfall Area – Tyrol, Workshop on Advances Techniques for the Assessment of Natural Hazards in Mountain Areas, Austria, 2000.

Simon, D., Kalman filtering for fuzzy discreet time dynamic systems, Applied soft computing 3(3), 2003.

Stefik, M., Introduction to knowledge systems. 2nd Edition, Morgan Kaufmann, San Francisco, 1998.

Subbu, R. and Sanderson, C., Network-based distributed planning using coevolutionary algorithms, Intelligent Control and Intelligent Automation, Vol. 13, World Scientific Publishing Company, 2004.

Thienelt, M., Eichhorn, A. and Reiterer, A., Intelligent pedestrian positioning in vienna: Knowledge-based Kalman-filtering (wikaf), ISPRS Proceedings, Vol. XXXVI, Part 5/C55 (2008), 315–321.

Vicovac, T., Reiterer, A., Egly, U., Eiter, T. and Rieke-Zapp, D., Intelligent deformation interpretation, in: Proceedings of Second International Workshop (AIEG 2010) – Application of Artificial Intelligence and Innovations in Engineering Geodesy, Braunschweig, Germany (2010), 10–20.

Wieser, A., Robust and fuzzy techniques for parameter estimation and quality assessment, Shaker Verlag Aachen, 2002.

Williams, R. J., Training recurrent networks using the extended kalman filter, in: Proceedings of the IJCNN'92 Baltimore 4 (1992), 241–246.

Zimmermann, H. G., Fuzzy sets theory and its applications, Kluwer, Nijhoa, Boston, Dordrecht, Lancaster, 1985.

Received: Sep 24, 2010 Accepted: Oct 27, 2010

Author Information
Alexander Reiterer, Tanja Vicovac
Institute of Geodesy and Geophysics
Vienna University of Technology, Austria
E-mail: alexander.reiterer@tuwien.ac.at, tanja.vicovac@tuwien.ac.at

Uwe Egly
Institute of Information Systems
Vienna University of Technology, Austria
E-mail: uwe@kr.tuwien.ac.at

Enrico Mai
Department for Geodesy and Geoinformation Science
Technische Universität Berlin, Germany
E-mail: enrico@mca.bv.tu-berlin.de

Shahram Moafipoor
Geodetics Inc.
San Diego, CA, USA
E-mail: smoafipoor@geodetics.com

Dorota A. Grejner-Brzezinska
Department of Civil and Environmental Engineering and Geodetic Science
The Ohio State University, USA
E-mail: dbrzezinska@osu.edu

Charles K. Toth
Center for Mapping
The Ohio State University, USA
E-mail: toth@cfm.ohio-state.edu