3D Scientific Data Interpretation using Cooperating Agents

R. J. Gallimore [1], N. R. Jennings [2], H. S. Lamba [1], C. L. Mason [3], B. J. Orenstein [1]

[1] BHP Research, Newcastle Laboratories, PO Box 188, Wallsend, NSW 2287, Australia.

[2] Department of Electronic Engineering, Queen Mary and Westfield College,
University of London, London E1 4NS, UK.
n.r.jennings@qmw.ac.uk

[3] Department of Electrical Engineering and Computer Science, University of California,
Berkeley 94720, USA.

Abstract – Many organisations collect vast quantities of 3D scientific data in volumetric form for a range of purposes including resource exploration, market forecasting and process modelling. Traditionally, this data has been interpreted by human experts with only minimal software assistance. However such manual interpretation is a painstakingly slow and tedious process. Moreover, since interpretation involves subjective judgements and each interpreter has different scientific knowledge and experience, formulation of an effective interpretation often requires the co-operation of numerous such experts. Hence there is a pressing need for a software system in which individual interpretations can be generated automatically, and then refined through the use of co-operative reasoning and information sharing. To this end, a prototype system, named *SurfaceMapper*, has been developed in which a community of co-operating agents automatically locate and display interpretations in a volume of 3D scientific data. The challenges and experiences in designing and building such a system are discussed.

1. INTRODUCTION

Many organisations collect vast quantities of 3D scientific data in volumetric form for a range of purposes including exploration, market forecasting and process modelling. These datasets are typically of the order of gigabytes although in some cases they may be terabytes. The storage, processing and interpretation of such large volumes of data is a major challenge. Indeed it is not uncommon for large historical datasets to lie unexamined simply because there are insufficient resources available to process them. However, since the datasets may contain valuable information that could lead to major opportunities, many organisations are investigating the use of advanced information processing techniques for this domain. Here we report on our experiences with the application of one such technique – cooperating agents – to a common problem faced by The Broken Hill Proprietary Company (BHP) [1] in the area of interpreting large volumes of 3D scientific data.

In our application, interpretation of 3D scientific data requires the identification of the size, location and orientation of 3D surface features in a data volume. The interpretation process is a complex information processing activity that has a significant subjective component. Each expert has an area of specialisation and many years of experience in forming interpretations.

Nevertheless, different experts often arrive at different interpretations of the same dataset (because they may have different levels of experience or skill bases). These differences in interpretation may dictate different follow-up actions which, in turn, could have widely varying financial implications. For this reason, domain experts often collaborate with one another to compare their interpretation hypotheses. Through this sharing of initial hypotheses, the experts refine their interpretations: They therefore need to create and delete surfaces; split and merge surfaces; modify the size, shape, location and orientation of surfaces; and then amend their confidence ratings in their interpretations. In some cases, this sharing of results brings the experts closer to a consensus; whereas in other cases, the experts continue to hold significantly different opinions.

The sheer size of the datasets and the amount of interaction required between domain experts, means that manual interpretation is a painstakingly slow, tedious and expensive process. Owing to the complexity of the interpretation task, there is also no guarantee of the completeness or accuracy of the manual interpretations. Support for automated interpretation is therefore desirable. At the time of writing, there are only a few commercial software tools that can assist domain experts in the formation of interpretations (e.g. AVS [2], Iris Explorer [3] and Khoros [4]). However these tools have limited flexibility, require significant input and direction from the user, and do not support collaboration between the experts.

Given the limitations of the currently available software tools, it was decided to develop an automated interpretation support system named *SurfaceMapper*. Practical experience from the domain indicated that the system should not take the form of a single analytical technique. Many organisations have tried solutions founded on single techniques including statistical methods, image analysis, pattern recognition, expert systems, neural networks and fuzzy logic – and have come to the conclusion that there is no broadly applicable best method or technique. Each one has its relative strengths and weaknesses. Some can only produce an approximate solution, but do so comparatively quickly; others are more accurate, but relatively slow. Furthermore, a given technique's performance is often dependent on the nature and signal to noise ratio of the data set. It is often impossible to determine *a priori* which technique is the most appropriate for a given volume or a given portion of a volume. Hence the system needs to be responsive to its problem solving context. Having identified the most promising areas of the data, it must dynamically select the technique which is best suited to the characteristics of that data (i.e. the data analysis system as a whole needs to be context sensitive or *situation-based*).

To overcome the problems associated with selecting from a number of independent techniques, some organisations have constructed tools which allow multiple methods to co-exist. However such tools typically place a significant burden on the user. For each technique, the user is expected to know its problem solving characteristics, be able to judge when, where and how to apply it, and to determine how best to fuse the results it produces. Our approach is to provide a wide range of uncoupled base techniques and to allow the software to determine at run-time, which technique is appropriate for use (or reuse) in particular circumstances. Moreover, the interchange and cross-checking between the techniques are required to be directly supported at the software level.

When taken together, the requirements for: (i) multiple interpretation techniques to co-operatively interwork and (ii) situation-based selection and execution of the different interpretation techniques, mean that a multi-agent approach is the most natural means of modelling and implementing the system. In our case, a multi-agent system is one in which the

key abstraction used is that of co-operating agents. By an agent, we mean an encapsulated software entity which exhibits the following characteristics [5]:

- *autonomy*: operates without the direct intervention of humans or other agents, and has control over its actions and internal state;

- *responsiveness*: perceives its environment (which may be the physical world, a user, a collection of agents, etc.) and responds in a timely fashion to changes that occur in it;

- *proactiveness*: does not simply act in response to its environment, but exhibits opportunistic, goal-directed behaviour and takes the initiative where appropriate;

- *social ability*: interacts, when it deems appropriate, with other agents in order to complete its own problem solving and to help others with their activities.

Given this paradigm, each interpretation technique can be considered as an autonomous agent which co-operates with other interpretation technique agents in order to try and converge on a community-wide solution. However, in many cases the agents are simply not able to achieve such a consensus because the interpretations of the individual agents vary significantly. This is especially true when the dataset is noisy. Thus co-operation can be viewed as providing the *opportunity* for the agents to converge. However failure to converge does not mean that the system's results are unusable. In these circumstances it actually mirrors the situation of the disagreeing human interpreters. In this latter case, differing opinions produced by the system enable the human expert to experiment with a number of what-if scenarios (calculating costs and likely revenue generation) based on a range of plausible interpretations.

The co-operating agents approach offers a number of advantages over the single solution technique and the multiple co-existing techniques approaches described earlier. Firstly, interpretations can be made more reliable through the automatic cross-checking of results. Secondly, early solutions can be presented to the domain expert (through selection of quick approximate techniques) and these can then be incrementally refined for as long as is deemed necessary (as the more accurate but time-consuming techniques produce their results). This means the overall system has an anytime capability. Finally, co-operation provides the opportunity to achieve an overall improvement in the quality of the interpretations. Presently there are a number of interpretation algorithms which give good results but which cannot be used because they examine the data in minute detail (and hence are simply too computationally expensive for any large data volumes). However if these techniques are seeded with results provided by computationally cheaper techniques, then the reduced search space means that they become applicable and can be used in promising, but smaller, regions of the data volume.

This paper makes contributions in two distinct areas. Firstly, it describes a novel approach to designing and building 3D scientific data interpretation systems. The insights and experiences gained in this work will enable other researchers faced with similar problems to assess whether an automated approach based on co-operating agents is appropriate for them. Secondly, this paper advances the state of the art in multi-agent systems: it provides an indication of the types of co-operation which occur in an important class of real-world applications, and it indicates how agent systems can be designed to cope with vast quantities of data.

The remainder of this paper is structured as follows. Section 2 introduces the basic terms and concepts of 3D scientific data interpretation. Section 3 describes the top-level design of the

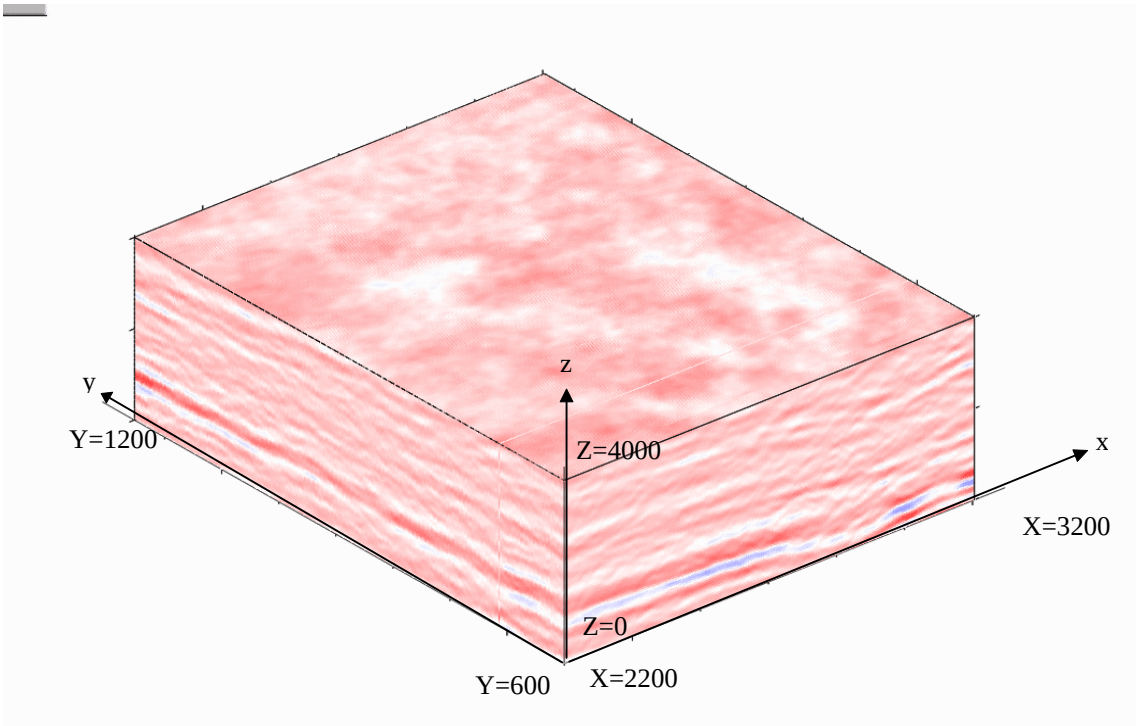

FIGURE 1. A 3D Scientific Data Volume.

SurfaceMapper multi-agent system and section 4 explains the architecture of the individual agents. Section 5 details the various co-operative scenarios present in SurfaceMapper. Section 6 discusses related work to place this research in context. Finally, section 7 presents our conclusions and highlights areas of further investigation.

2. DOMAIN TERMS AND CONCEPTS

This section provides an introduction to the terms and concepts of 3D scientific data interpretation. Our 3D scientific data set may be depicted as a volume consisting of 1000 x 600 x 4000 voxels (figure 1). This data set is typical of volumetric data sets which are obtained by remote sensing techniques. In such techniques, sensors are located outside the control volume and information relevant to the inside of the volume is inferred from the sensed signals. Often, these data sets do not have perceived characteristics or bounding conditions, and the interpretation may not be definitive.

A 2D cross-section through this volume (figure 2) is called a *slice*. An interpretation may be made for a specific slice of the data or for the entire volume. On a particular slice, the features of interest are piecewise linear 2D *segments* (henceforth called *segments*) and connected 2D *curves* (henceforth called *curves*). Segments represent edges or lines of varying clarity identified by individual or multiple analytical techniques within a single slice. In the volume, the features of interest are the 3D surfaces (henceforth called *surfaces*).

The objective of SurfaceMapper is to identify the most relevant *surfaces* in the data volume. A relevant surface is one which satisfies certain typical domain characteristics such as minimum size, expected orientation, expected shape, etc. To achieve this objective in an automated system a number of basic steps need to be performed. Firstly, *segments* need to be identified in

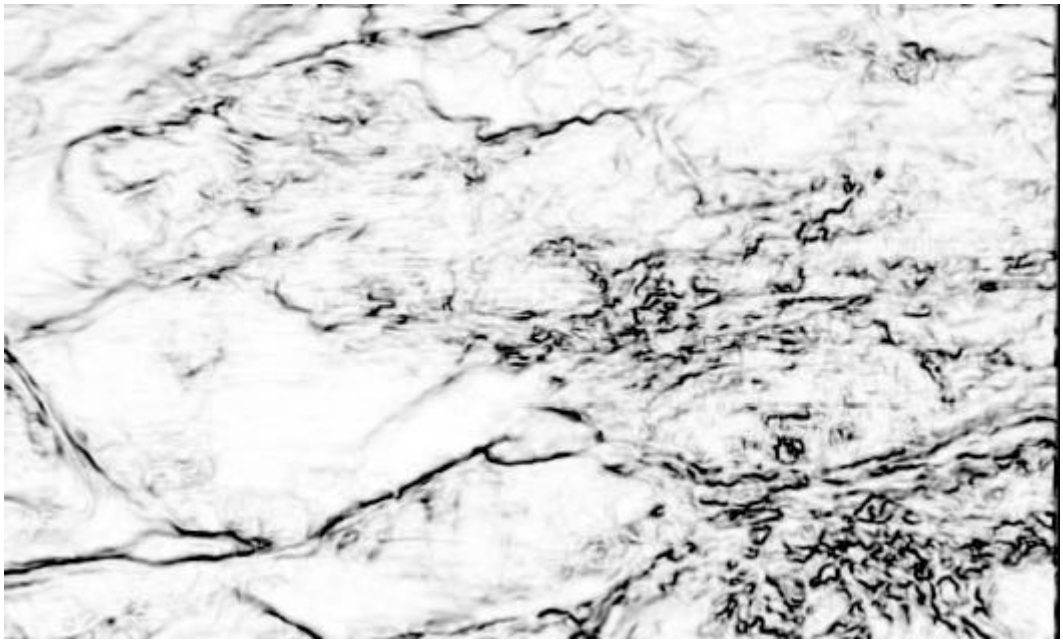

FIGURE 2. A specific data slice through the 3D volume at z = 912.

individual slices (figure 3). These *segments* then need to be associated together to form *curves* (figure 4). Finally, the association of *curves* across multiple slices results in the formation of *surfaces* (figure 5).

3. THE SYSTEM ARCHITECTURE

SurfaceMapper was designed with sound software engineering principles in mind. The system was therefore decomposed into cohesive, minimally coupled, modular subsystems. This approach reduced the overall system complexity, has facilitated modification and maintenance, and encouraged parallel development of the individual subsystems. Following these basic principles, SurfaceMapper was divided into four subsystems: the scientific data subsystem (section 3.1), the analytical techniques subsystem (section 3.2), the agents subsystem (section 3.3) and the user interface subsystem[1].

SurfaceMapper's design achieves high cohesion and low coupling through partitioning into logical subsystems with simple interfaces between them (figure 6). The scientific data subsystem abstracts all data specific details (e.g. data formats, dynamic range details and floating point to integer conversions) and provides access to any specified slice of data. The analytical techniques subsystem encapsulates algorithmic details and provides access to segments extracted from a specified slice. The agents subsystem uses co-operative reasoning to associate segments into curves and curves into surfaces. It also passes these interpretations to the user interface subsystem. All end-user interaction and visualisation responsibilities are

[1] Given the space limitations we are unable to describe the user interface sub-system in detail. However we note that it has two roles: visualisation and end user (domain expert) interaction. The visualisation component shows the operation of the system. It can be used for development and debugging purposes, as well as for indicating the way SurfaceMapper is working to the domain expert when the system is in operation. The user interaction component displays the system's interpretation of the dataset to the domain expert. The user can turn individual interpretations on or off, view the interpretations in 2D or in 3D, zoom in or out as desired, and rotate the volume to obtain the best view.

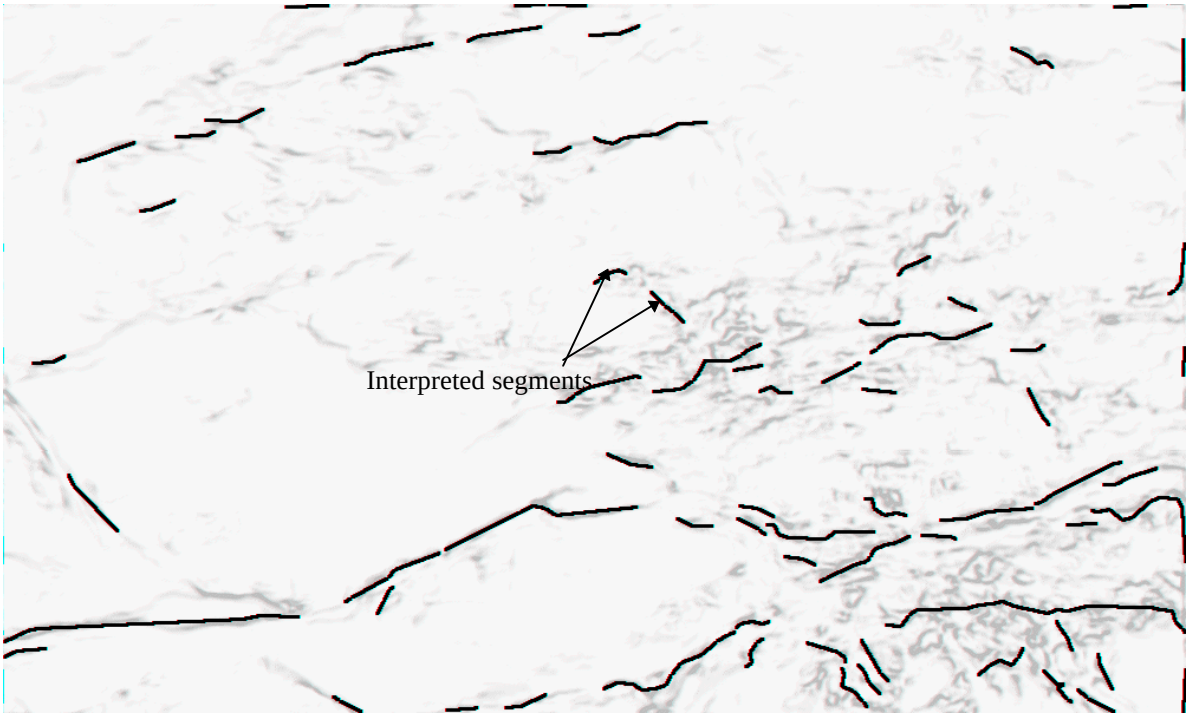

FIGURE 3. Interpreted Segments in Data Slice at z = 912.

handled by the user interface subsystem. This subsystem displays the final interpretations to the user, presents a view of the system's ongoing operation, and allows testing and debugging operations to be performed.

This system design is typical of the class of distributed problem solving (DPS) applications. Multiple agents cooperate by dividing and sharing knowledge about a common problem and its solution [6]. DPS systems have three phases: (i) task decomposition and assignment, (ii) solution of the individual sub-tasks, and (iii) synthesis of results. In SurfaceMapper, decomposition and assignment are carried out at design time (as indicated by figure 6) and the solution and synthesis phases predominate at run-time. In order to solve their individual sub-problems and synthesise their results, SurfaceMapper's agents exchange what they consider to be useful and relevant information with one another. That is, they exhibit a *result sharing* form of co-operation [7]. This shared information is used by the recipient to alter its confidences in its interpretation hypotheses or to focus its problem solving in promising regions of the search space.

3.1. The Scientific Data Subsystem

The scientific data subsystem provides access to all the slices in the volume. For our purposes, these slices represent the raw data from which the interpretations are made. The slices may vary in two aspects. Firstly, they can be horizontal, vertical or oblique slices through the volume. Secondly, they can be gathered using a number of different techniques. These different types of data provide complementary representations of the same 3D volume.

Specialised algorithms are required to process the different data types and to make interpretations of slices in varying orientations. In SurfaceMapper only algorithms for processing horizontal slices have currently been implemented, and only one data type is

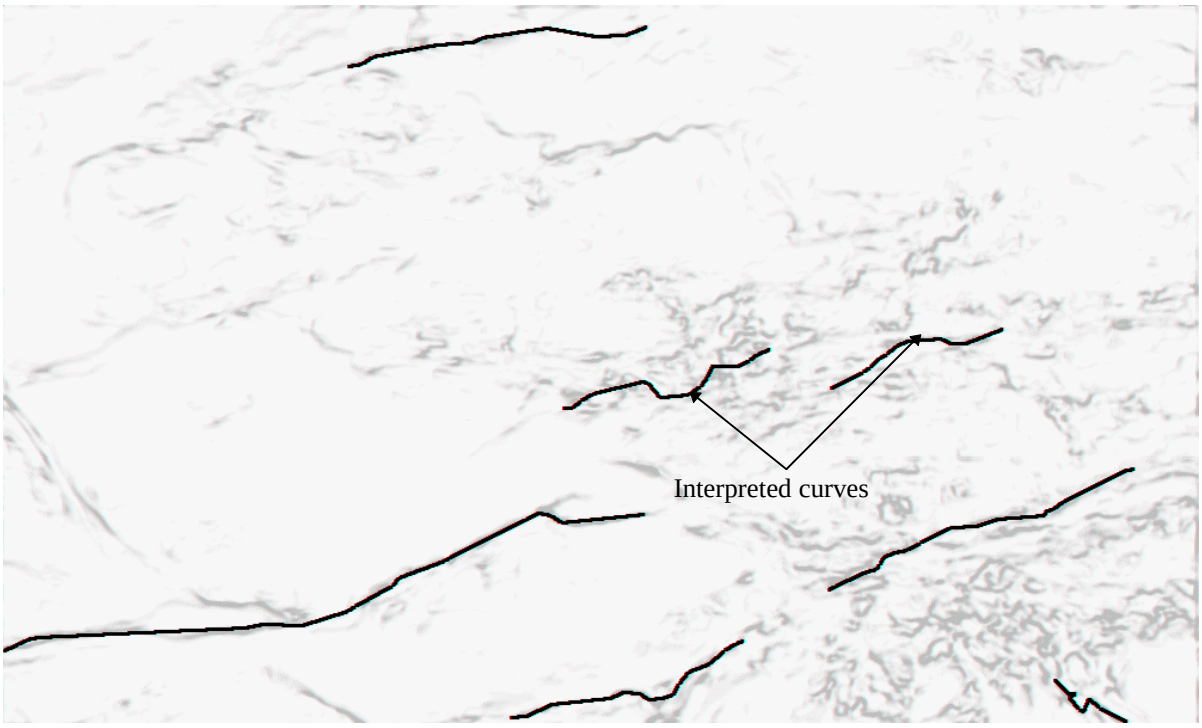

FIGURE 4. Interpreted Curves in Data Slice at z = 912.

currently supported. However, further algorithms which operate on vertical slices and/or on multiple data types can be added without changing the system architecture.

3.2. The Analytical Techniques Subsystem

The analytical techniques subsystem contains a number of analytical image processing algorithms that extract *segments* from the slices in the scientific data subsystem. These algorithms are proprietary developments which are typically based on line detection algorithms with derivative edge masks. The algorithms vary in the time taken to produce a solution and the quality of the solution produced. Some algorithms give quick approximate solutions, while others give more accurate solutions but usually require significantly longer computation time.

Each algorithm follows the same basic sequence of steps, although the techniques and representations vary. The steps include pre-processing techniques for image enhancement, edge and line detection techniques, line linking and consistency checking. The algorithms also provide an objective rating (common across all techniques) of the quality of the segment that is used to seed the processing in the agents subsystem. In this case, “high quality” means highly linear segments (i.e. those which have a high ratio for the shortest distance between their end points and their actual curvilinear length) from a low noise data section. An entropy measure is used to estimate a slice’s noise content.

3.3. The Agents Subsystem

Like many data interpretation processes [8], the task of interpreting 3D scientific data can be described in terms of a signal to symbol transformation, where raw data is transformed into a high level concept by following a natural data abstraction hierarchy. As described in section 2, surface interpretations are produced from curve interpretations which, in turn, are produced from segment interpretations that are seeded by basic segments formed in the analytical

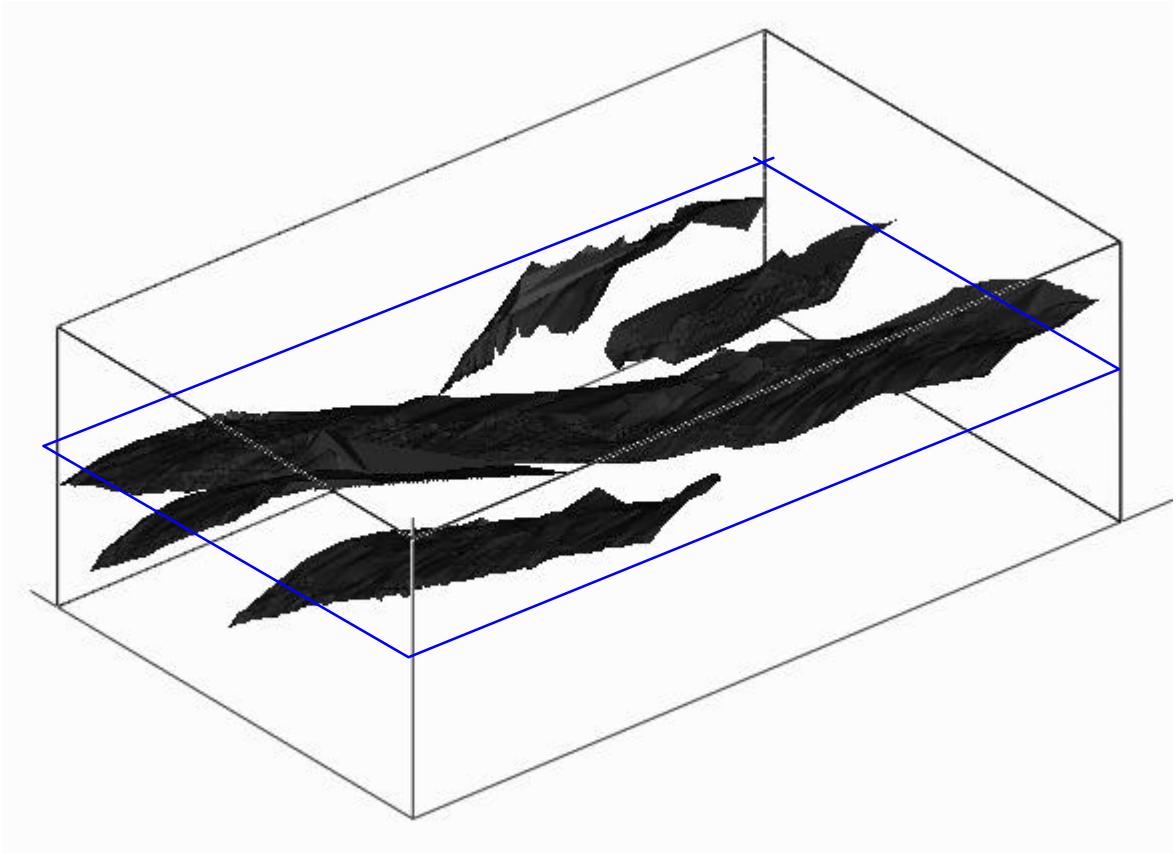

FIGURE 5. Interpreted Surfaces in the 3D volume

techniques subsystem. It was decided that the most natural means of partitioning SurfaceMapper's agent subsystem was according to these three levels of data abstraction: surfaces, curves, and segments. Each partition, or *agency*, is composed of a number of concurrently operating agents, each possessing complimentary expertise or perspectives. The control structure within an agency is that of a committee of *peers* with no central authority. Communication of interpretations occurs both within an agency (section 3.3.1) and between agencies (section 3.3.2) and although individual agents strive for agreement, it is by no means a solution requirement (see section 1).

In general, SurfaceMapper agents strive to provide independently derived interpretations. However, where it enhances their problem solving performance – either by increasing their confidence in their interpretation or by directing their search to promising regions of the data volume – they exploit hypotheses generated by other agents. This structure and mode of operation offers three main advantages. Firstly, it provides *composability* – the ability to plug-in or remove various interpretation techniques (formulated as agents) as new algorithms are developed. Secondly, it provides *algorithm opportunism* – the ability to dynamically select interpretations from the agents whose approach yields the best solution. Finally, it allows *solution multiplicity* – the ability to present the domain expert with a number of reasonable interpretations. The presence of multiple interpretations is also an important message to the user as it allows a number of follow up scenarios to be investigated.

We refer to the intra-agency distributed problem solving paradigm as *Functionally Independent/Co-operative* (FI/C) after Lesser's Functionally Accurate/Co-operative (FA/C) [6] paradigm (refer to section 6 for a more detailed discussion and comparison of the two approaches). "Functionally independent" describes the situation where agents build

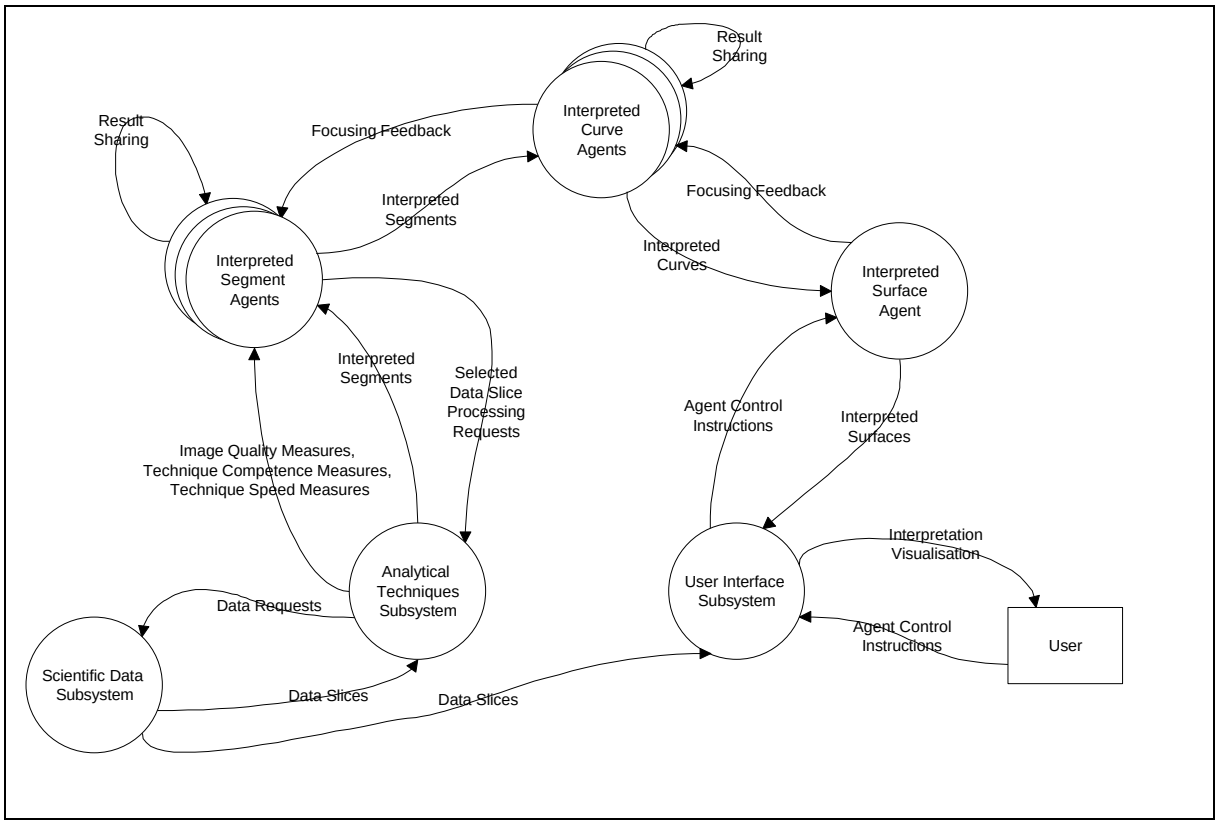

FIGURE 6. SurfaceMapper System Architecture.

independent, non-convergent solutions though the exchange of partial solutions which may be inaccurate or inconsistent. “Co-operative” describes an interactive exchange of heuristics that can guide an agent’s independent search for a solution. The overall effect of SurfaceMapper’s FI/C approach is that an agency as a whole can dynamically exploit the most suitable techniques for a particular data instance – reflecting the fact that no single technique is superior to all others in all cases.

In the remainder of this section we describe each of the three agencies in terms of their roles in the system and how they are realised. We then we discuss the inter- and intra- agency interactions.

The segment agency is responsible for exploiting the services of the analytical techniques subsystem to produce segments for any desired slice. The most promising segments are then provided to the curve agency in a proactive or reactive manner. Essentially the segment agents are intelligent wrappers around the analytical data processing techniques (each technique being associated with precisely one agent). What the agents add to the processing undertaken by the analytical techniques is the ability to filter and rank promising segments generated by the analytical techniques subsystem. They do this by exploiting a set of simple domain-dependent heuristics. For example, an agent will have high confidence in both low curvature segments that come from a low noise area and medium length segments. Very short segments are typical of small isolated noise patches (see the top right-hand corner of figure 2 and the corresponding short segments in figure 3). Very long segments are typical of a dense collection of noise patches (see the lower right-hand corner of figure 2 and the segments extracted from that area in figure 3).

To develop preliminary hypotheses about the most prominent segments in the volume, a segment agent identifies those slices that have low noise content. This is done by computing the entropy of each slice and selecting those slices which have the lowest value. The segment agent then asks its associated analytical technique to process these slices. This provides a series of slice interpretations containing the identified segments along with an indication of their perceived quality. Depending on the quality and quantity of segments received, the segment agent may chose to modify the operating parameters of the analytical algorithm to make it generate segments which are of more appropriate length, of higher confidence, or of greater linearity. To increase its confidence in an interpretation of a particular slice, a segment agent may examine the neighbouring slices (above and below). If a match is found between segments on neighbouring slices, then the agent increases its confidence in its interpretation of that slice. The intra-agency interactions centre on requests and exchanges of segment interpretation hypotheses in order to improve the quality of the solutions produced by the individual agents. Agents increase the confidence of those interpretation hypotheses which agree with hypotheses sent by their peers.

The *curve agency* is responsible for constructing curves by associating related segments within slices. The cross-section of a surface on a slice appears as a series of disconnected collinear segments. Hence, agents in the curve agency have to identify those segments in the slice which are part of a 3D surface and associate them into a 2D connected curve. The most promising curves are then provided to the surface agency in a reactive and/or proactive manner. A curve agent typically receives a large number of disconnected segments from one or more segment agents. Some of these segments will be part of a surface, while others will be artefacts caused by noise. To reject the artefacts, the curve agents employ a number of domain specific heuristics. For example, a segment is considered more likely to be spurious if: i) it has been given a relatively low confidence by the segment agency; ii) it is a short segment which has no collinear relationship with any other segment; or iii) there are no similar segments identified in the neighbouring slices. Having filtered spurious segments, a curve agent forms curves from segments which have similar orientation and have endpoints which are close to each other. The confidence of this curve is determined from the average confidence of the constituent segments and on the number of matching curves (i.e. curves with similar length, location and orientation) generated by relevant *hint resources*. For a curve agent C processing a slice S, a hint resource may be any one of: i) a neighbouring slice above or below S, processed by the same curve agent C; ii) the same slice S processed by a peer agent; or iii) a neighbouring slice processed by a peer agent. The operational semantics of this broad pattern of behaviour vary between curve agents which, allied with the fact that they are working with different segment interpretations, means the curve interpretations can vary significantly. For this reason, curve agents share curve interpretation hypotheses (as discussed above for segment agents) in order to try and reach a consensus. As before, failure to converge on this level means that there is more than one plausible interpretation of the data volume.

The *surface agency* is responsible for constructing 3D surfaces by associating related curves across multiple slices. The agent forms a surface where it finds a series of similar curves in successive slices. Curves that come from neighbouring slices and have similar location and orientation are associated together to form a surface. The confidence of this surface is computed by averaging the confidence of each curve in the surface. In the current version of SurfaceMapper, there is a single surface agent. This agent computes surfaces in one of two ways: (i) based on a single curve agent's output; or (ii) by fusing together the curves produced by multiple curve agents. If all the curve agents are considered equally competent for the

dataset being processed then option (ii) is exercised, else option (i) is exercised and the agent chooses what it considers to be the most competent agent. Interpreter direction occurs when the domain expert wishes to experiment with numerous what-if surface interpretation scenarios. This is akin to asking multiple human interpreters and receiving different opinions.

3.3.1. Intra-Agency Interactions

Agents in all of the agencies conform to the same basic model of problem solving behaviour. Each agent uses its unique problem solving expertise (e.g. about producing segments, curves or surfaces) to produce an independent opinion about the solution to the interpretation task at hand. In so doing, the agent may exploit information it has generated about interpretations from neighbouring slices. The agent's opinion consists of a ranked list of hypotheses about the likely interpretation of the data being processed.

If an agent is confident about its opinion then it proactively volunteers it as a hint to its peers (which is an example of *data-driven result sharing*). Upon receipt of a hint, an agent may increase its confidence in a hypothesis that agrees with the received information or it may generate new hypotheses based on this new information (see section 5 for more details). In either case, the agent may decide that a further round of domain problem solving is needed (to take account of the new ranking or to investigate the new hypotheses) or that it should volunteer new hints to its peers. This form of unsolicited result sharing terminates when an agent has no new high confidence opinions to volunteer to its peers.

The other form of intra-agency interaction is *demand-driven result sharing* - which is the key approach used in SurfaceMapper. This involves an agent actively seeking the opinion of one of its peers (i.e. results are explicitly asked for rather than spontaneously volunteered). This happens when an agent has a number of interpretation hypotheses which have low confidence values or when the agent wishes to start its processing in a promising region of the search space (rather than starting from scratch and systematically working through the whole data set). This form of result sharing requires the agents to maintain an acquaintance model of each of their peers so they can reason about the most relevant one to take hints from. This reasoning is based on competence information – for a given type of data, the acquaintance model indicates which agents are considered fast, which are slow, which are precise, and which are inappropriate. Having selected an acquaintance, the agent asks it for its current interpretation opinions. When these opinions are returned to the request originator, they are used to change the confidence of existing hypotheses or to construct new hypotheses. An agent will stop soliciting hints from its peers when the information received produces no significant hypothesis changes or when conflicting peer opinions result in successive positive and negative reinforcements (thus causing confidence oscillations).

Agents in an agency benefit from each others' expertise and opinions, but do not have to arrive at a consensus. This models the process of human interpretation in which interpreters benefit from discussing their interpretations with one another, but they do not necessarily arrive at a consensus. It is up to the agent consuming an agency's results to choose whether and how best to combine the differing opinions. In the case of SurfaceMapper, this fusion takes place both within the curve and the surface agencies. When a curve agent receives segments from more than one segment agent, it computes its own confidence value for each segment. This value is based on the curve agent's assessment of the competence of the agent that provided the segment (stored in the curve agent's acquaintance model) as well as the segment agent's own confidence in its interpretation. At this point, the curve agent has a collection of segments,

together with its rating of their confidence levels, from multiple segment agents. For segments which have similar orientations and endpoints close together, the interpretations are merged and the confidence rating of the segment is raised accordingly. For segments which are identified by only one agent, the confidence value is left unchanged. Having merged all the similar segments, the curve agent now has a collection of segments, all of which are dissimilar. These unique segments are, where appropriate, associated into curves. Similarly, when a surface agent receives curves from more than one curve agent, it modifies the confidence of the curve interpretation based upon its perception of the competence of the curve agent that supplied it, merges the similar curves, and then forms surfaces from these curves.

3.3.2. Inter-Agency Interactions

As the agencies are arranged in an interpretation abstraction hierarchy, inter-agency interaction is an inevitability. This structure was chosen because there are numerous ways in which the data at one level can be used at the neighbouring levels. Each higher level agent is provided with a choice of results from a pool of resources each with different characteristics and properties. However if the system is to operate effectively, these interactions need to be co-ordinated to ensure that there is an appropriate flow of information which occurs in a timely and ordered fashion.

In this context, there are two primary forms of result sharing: (i) *result feedforward* in which information flows from the lower to the higher level agencies; and (ii) *focusing feedback* in which information flows from the higher to the lower levels. An example of the former is where a segment agent passes valid high-quality segments to a curve agent. An example of the latter is where a curve agent, having formulated some preliminary hypotheses, requests more (possibly lower quality) segments from the segment agent. The objective of such a request may be to try and extend an existing curve so this curve is provided to constrain the area in which the segment agent's search should take place. More details of both types of interaction are given in section 5.

4. THE AGENT ARCHITECTURE

All SurfaceMapper agents have the same architecture. This architecture is a belief-desire-intention (BDI) architecture and is characterised by having the mentalistic notions of beliefs (what the agent knows about its environment), desires (states the agent wishes to achieve), and intentions (selected courses of action) at its core. Such an architecture was chosen because it has a proven track record of working effectively in a range of complex applications – including air traffic control [9], process control [10], simulation [11], fault diagnosis [12] and transportation [13]. The specific BDI implementation used in this work is dMARS [14] (figure 7).

Each dMARS agent has a set of desires, goals, beliefs, plans and intentions. It receives percepts from its operating environment and performs actions to affect its environment. Agents use messages to communicate and share results with one another. SurfaceMapper uses the following speech-act based primitives for intra- and inter- agency based interactions: ASK (to request information from another agent—used for initiating demand-driven result sharing and focusing feedback), REPLY (to respond to an ASK message) and TELL (to volunteer information to another agent—used for initiating data-driven result sharing and result feedforward).

The operation of the agent architecture is best explained by means of an example. Consider the behaviour of a curve agent (left hand side of figure 7). An agent has a set of inbuilt (fixed) desires which specify its objectives. Of these desires, some may be chosen to be pursued in the current context (while the others remain dormant). This subset of active desires corresponds to the agent's current goals. For example, the curve agent might have a goal to link segments into curves and a goal to communicate high confidence curves to its peers. To be able to satisfy these goals, a curve agent needs to maintain certain beliefs. These include: the acquaintance models of its peers (which contains information such as algorithm speed, level of detail, and reliability), information about segments received from the segment agents (e.g. endpoints, length, linearity and entropy), and information about curve hints received from peer agents and neighbouring slices (e.g. curve endpoints, curve confidence and curve angle).

For each goal the agent can satisfy, it has an associated set of plans (stored in a plan library) that may achieve the desired objective when invoked. For instance, one plan may allow the agent to form curves from segments collected from a segment agent, another plan may allow the agent to form curves with the aid of hints collected from one of its peers, while yet another may allow the agent to form curves with the aid of hints collected from neighbouring slices. Once an agent decides to run a plan to try to satisfy a goal, a corresponding intention is created. This intention persists until the agent believes the intention cannot succeed, until it is no longer deemed appropriate, or until it is satisfied. When an intention finishes (either by succeeding or by failing), the agent's goals and beliefs are updated.

An agent responds to events, which may either be messages received from other agents or goals established from within its own plans. The occurrence of an event results in the invocation of one or more plans to deal with it. Which plans get invoked is determined by the agent's interpreter and is based on the type of events that the plan responds to, and the beliefs that the agent must have for the plan to be applicable under the present circumstances.

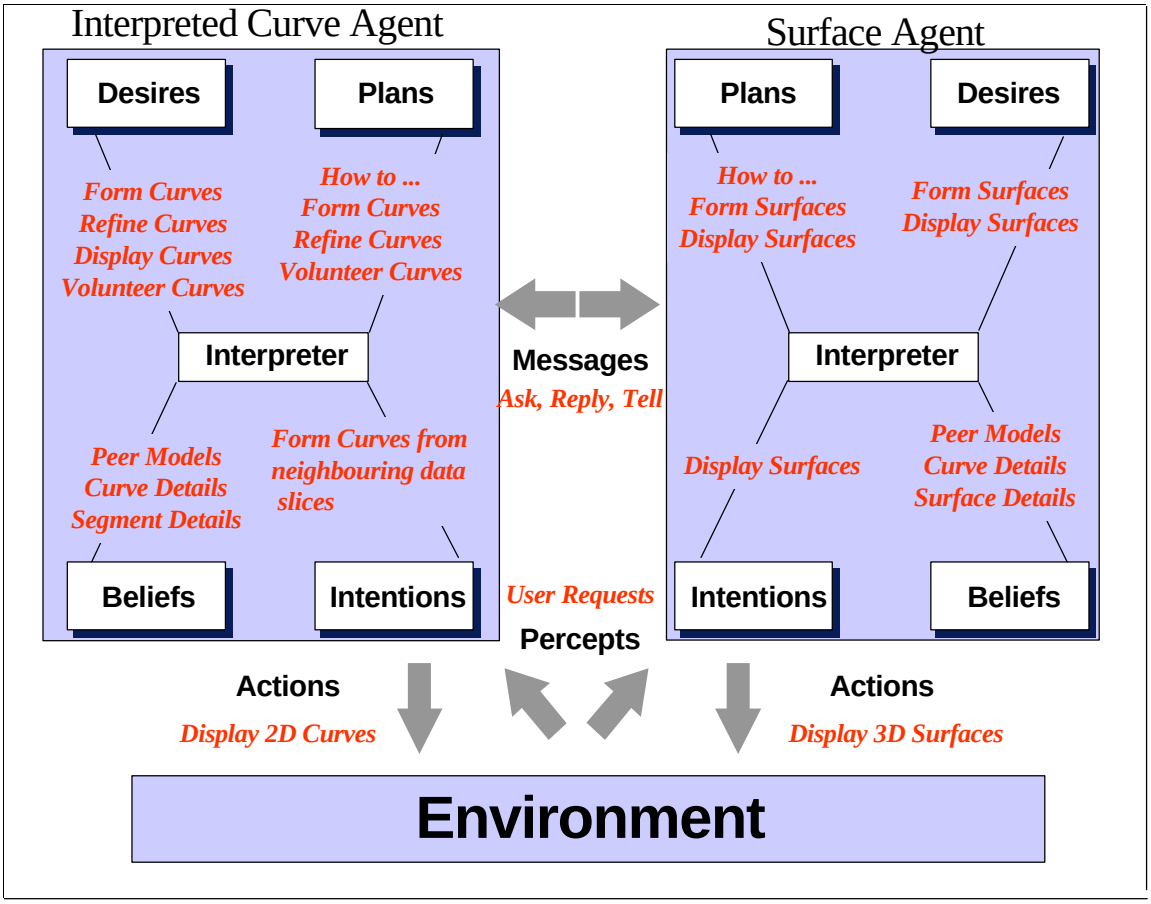

Figure 7. The dMARS Agent Architecture.

5. COOPERATION SCENARIOS

SurfaceMapper uses three types of result sharing co-operation: result feedforward, intra-agency result sharing and result feedback (section 3.3). In feedforward mode, an agent passes its results to an agent in the next higher agency. For example, a segment agent may pass the validated segments from a particular slice to a curve agent. With intra-agency result sharing, an agent passes its results to its peers. For example, a curve agent may pass its curve hypotheses to a peer agent, to assist the peer in refining its hypotheses. In feedback mode, an agent passes its results to an agent in the next lower agency. For example, a curve agent may pass its curve results to a segment agent to help the segment agent focus its validation criteria in the region close to the indicated curve results.

In all cases, co-operation may be initiated by an agent volunteering information it believes to be relevant to its acquaintances or by an agent making an explicit request for the information it needs. When an agent has high confidence hypotheses, it may voluntarily choose to forward these results to other agents. Hence a curve agent that has just found a curve with very high confidence would send this curve to its peers to assist them in their search for curves (intra-agency result sharing), to a segment agent to allow it to find valid segments in the neighbourhood of the identified curve (result feedback), and to the surface agent to allow it to form surfaces from high confidence curves (result feedforward). In other cases, an agent may request hypotheses about a specific level of interpretation from another agent in order to gather input to enable it to start processing or to refine its current hypotheses.

In more detail, co-operation between agents and agencies occurs at four distinct levels:

1. *Co-operation between segment agents*. Two segment agents may apply different validation criteria on segments extracted by two different analytical algorithms from the same slice. Having applied their individual criteria, each agent has its own hypotheses on the likely segments in the slice. These agents may then exchange their hypotheses (voluntarily for high confidence hypotheses, and on-demand otherwise), to help each individual refine their hypotheses based on the observations of their peers. For instance, if segment agent A volunteers information about a high confidence segment (SA) in a location close to segment (SB) found by segment agent B, then B will increase its confidence in SB by an amount proportional to A's confidence in SA.

2. *Co-operation between the segment agents and the curve agents*. A curve agent may ask a segment agent to produce and send it segments from a particular slice. The curve agent then associates those segments which are related into curves. Also if a curve agent forms some high confidence hypotheses about curves in a particular slice, it may provide them to a segment agent as focusing feedback. The segment agent could use the hypotheses it receives as an indication of regions of the data where it is likely to find segments. Given this focus, the segment agent can perform a more detailed analysis of the identified area with higher likelihood of success.

3. *Co-operation between the curve agents*. Two curve agents may produce related, but different, hypotheses about curves in the same or neighbouring slices. They may voluntarily, or on-demand, share their results, to help refine their hypotheses. For instance, curve agent A may volunteer a high confidence hypothesis from a particular slice, and the receiving curve agent B may have no curve in that region. In this situation, B would typically loosen its algorithmic parameters (e.g. use lower confidence segments that are further apart) to try and construct a curve in the identified region. If B is able to achieve this, then the overall confidence in the interpretation in this region is increased.

4. *Co-operation between the curve agents and the surface agent*. The surface agent may ask a curve agent to provide curves from a particular set of slices, or alternatively, a curve agent may voluntarily send high confidence hypotheses to the surface agent. In either case, the surface agent associates approximately collinear curves into a surface. The surface agent may also reject those curves that have little association with other curves and suppress those surfaces that have very few curves (because in this domain a surface extends through many slices and should hence contain many curves).

6. RELATED WORK

There are a number of interpretation systems based upon the concept of co-operating agents and these are compared and contrasted with SurfaceMapper in this section. The broad issues on which such systems can be compared include the way in which they perform their problem decomposition, their subproblem solution, and their solution synthesis, as well as in how they apply their domain knowledge. Differences on each of these dimensions are intimately tied to the design chosen for intra-agency, inter-agency, and internal agent problem solving behaviour.

Pioneering work in distributed interpretation system design was performed by Lesser and Erman for the task of speech understanding, HEARSAY-II [15], and by Lesser and Corkill for the task of vehicle monitoring, DMVT [16]. In both of these applications, the presence of

overlapping, redundant sensing data forced a decomposition where the subproblem solution phase and the synthesis phase required consistency. When coupled with the fact that the knowledge for interpreting the sensed data in each agent was identical, it can be seen that redundant solutions were of no particular value and were, therefore, viewed as wasted computation. Also only singular solutions were required in both applications – multiple competing solutions had to be avoided and suppressed since they were not useful to the end user. For these reasons, both HEARSAY-II and DVMT paid considerable attention to the problem of avoiding redundancy since sharing redundant subproblem solutions could increase the amount of search needed to reach a conclusion without adding any value. That is, it was considered globally incoherent system behaviour. Given this view, both systems explored numerous problem decompositions and worked to limit needless search by using various methods for focus-of-control for limiting the amount of inter-agent communication. Such techniques included organisational structuring (providing a high level specification of how the multi-agent system solves its problems and an indication of the role of each individual agent within this structure) and the exchange of meta-plans (control level information about an agent's current priorities and activities).

In general terms, the basic structure and operation of HEARSAY-II and DVMT can be described as Functionally Accurate, Co-operative (FA/C) [6]. Functionally accurate refers to the notion that the agents converge on acceptable and accurate solutions although they may share inconsistent or inaccurate partial solutions. Co-operative describes an iterative co-routine interaction among agents in deriving solutions. The success of HEARSAY-II and DVMT demonstrates that the FA/C structure is a powerful paradigm for applications in which: i) convergence toward a single consistent hypothesis is a requirement for subproblem solution and solution synthesis; ii) multiplicity of subproblem solutions is viewed as redundant; and/or iii) end-users require presentation of a single solution.

Subsequent work on distributed interpretation by Mason [17], in developing NETSEA (network seismic event analyser) for the domain of verifying nuclear testban treaties, defined an alternative, but related, system design paradigm. Unlike the speech understanding or the vehicle monitoring domains, in monitoring for treaty violations there are multiple, non-overlapping data streams. Moreover each such stream requires distinct interpretation knowledge.Consequentially, the problem decomposition resulted in an agent that fully interpreted each data stream. Agents relied on communicated results as heuristics, and focus-of-control was accomplished using introspection, since the human experts in the domain relied on qualitative data analysis methods. Instead of requiring convergence, an important feature of the system solution was to preserve differences of opinion among the agents, since conflicts among monitoring reports were considered to be of great interest by the user. These requirements were clearly contrary to those assumed by the FA/C approach and so led to the identification of the FI/C problem solving structure (see section 3.3) in which the differences among the subproblem solutions (agents) are preserved and where there is no longer a demand to produce a singular consistent output from the solution synthesis phase.

The designs of SurfaceMapper and NETSEA differ from the FA/C solutions in that the structure of the domain involved subproblems that exhibited *inter-dependency*, rather than dependency. SurfaceMapper, HEARSAY-II, and DVMT all exhibit decompositions involving an overlap of sensed data – in SurfaceMapper more than one agent in an agency may explore the same slices. However, as with NETSEA, SurfaceMapper's problem domain involves teamwork between data analysts possessing distinguishable knowledge and expertise, and a

solution where the presence of disagreement provides valuable additional information for users who view the system as one of many tools in their much larger work process environment. Interestingly, both NETSEA and SurfaceMapper were user-centred designs, inspired by naturally distributed problem solving systems. As such, their structure (FI/C) gives rise to the phenomena that by introducing more agents with different kinds of expertise, we increase the likelihood of finding "correct" or "more complete" solutions. This feature in some sense allows us to finesse the usual scaling and composability problems present in many man-made, complex distributed applications.

7. CONCLUSIONS AND FUTURE WORK

This paper has described the rationale, design, and implementation of a multi-agent system for interpreting 3D scientific data. Our development of SurfaceMapper has demonstrated its capability of performing automated surface interpretation from 3D scientific data. Feedback obtained from interpretation experts indicates that the level of performance attained by SurfaceMapper in terms of accuracy and cost effectiveness is such that it is a valuable decision support tool for this domain. In more detail, when compared against a human interpretation expert on a real-world dataset, SurfaceMapper obtained a coverage (the proportion of all manual interpretations that are picked out by the automated system) of 67% and a precision (the percentage of automated interpretations that agree with the manual ones) of 80%. Our control single solution technique obtained respective figures of 75% and 55% for the same dataset. Note that although the coverage of the control technique is better this is only because false-positives are not penalised; thus the control algorithm simply highlights many more potential traces which means the traces of interest to the interpreter are more difficult to spot. In short, the signal to noise ratio of the control technique is poor.

SurfaceMapper achieves its performance and flexibility by exploiting the co-operating agents metaphor and by using the FI/C mode of system operation. Co-operating agents provide a logical means of automating current best practice in manual interpretation and offer benefits in terms of the quality of solutions which are produced. SurfaceMapper's use of the FI/C paradigm is also appropriate given the subjective and uncertain nature of the interpretation task in this domain. This work provides important practical insights into, and guidance for, theoretical research into both co-operative problem solving and FI/C systems. Study of SurfaceMapper's operation, supported by the evaluation, highlights the fact that agents operating in such a manner need to carefully manage their interactions and their local reasoning so that they do not needlessly distract one another by sending irrelevant suggestions. Moreover, the evaluation shows that agents need a means of tracking the benefits accrued from social interaction so that the resources they consume in this endeavour are commensurate with the improvement in performance which is achieved.

Careful examination of the requirements and behaviour of SurfaceMapper's problem solving components offers a strong justification for the choice of agents as the preferred system development paradigm. The components are autonomous in that they automatically filter data archives and report useful information without user intervention or guidance. They are responsive in that the selection and weighting of the different analytical approaches depends upon the nature of the data volume being processed. They are proactive in that they spontaneously volunteer unsolicited advice to agents which they believe will benefit from receiving it. They are social in that they cooperate with one another to cross-check their results and to indicate promising regions of the data that may be worth investigating in detail.

There are a number of issues which require further investigation in the next phase of the project. Firstly, a more comprehensive suite of analytical techniques is required. Techniques which deal with vertical slices and with data gathered using non-acoustic means are needed to cope with the range of datasets which can be encountered in this domain. Secondly, the agents should be able to adapt and learn from the social interactions they experience. Agents should be capable of learning which acquaintances give reliable results and then adapt their selection and fusion techniques appropriately. Presently, this information is hard-wired into the acquaintance models and into the agent's problem solving plans. Finally, the agents would benefit from a more elaborate means of tracking the information they volunteer or receive during result sharing. Presently, an agent's record of information shared is insufficient to indicate to its acquaintances that it has significantly revised a hypothesis and that they should, therefore, roll-back any calculations they have performed based upon its preliminary result. Our present thoughts on this matter are that such interdependencies can be best handled by incorporating a distributed truth maintenance system [18] into the agent architecture.

Acknowledgements: This work was supported by the Advanced Computing Applications Project (ACAP) at BHP Research, Newcastle Laboratories. We wish to acknowledge the invaluable efforts of our team members: Adrian Ugray, Tony Murnain, Hok Min Lie, Peter Gordon, Peter Wall, Keith Nesbitt and Lawrence Leung. We thank the Australian Artificial Intelligence Institute for their support with dMARS. Most importantly, we thank Peter Littlejohn and Gus Ferguson for their support as program and project champions, respectively.

REFERENCES

[1] The Broken Hill Proprietary Company (BHP). http://www.bhp.com.au/

[2] Advanced Visual Systems Inc. http://www.avs.com/

[3] IRIS Explorer. http://www.nag.co.uk/Welcome_IEC.html

[4] University of Pisa. http://fisica.difi.unipi.it/IntroSP2/html/khoros/khoros.html

[5] Wooldridge, M. J., and Jennings, N. R., (1995). "Intelligent Agents: Theory and Practice" Knowledge Engineering Review 10 (2) 115-152.

[6] Lesser, V. R. and Corkill, D. D., (1981) "Functionally Accurate, Co-operative Distributed Systems" IEEE Trans on Systems Man and Cybernetics 11 (1) 81-96.

[7] Smith, R. G., and Davis, R., (1981) "Frameworks for Co-operation in Distributed Problem Solving" IEEE Trans on Systems Man and Cybernetics 11 (1) 61-70.

[8] Jagannathan, J., Dodhiawala, R., and Baum, S. (eds) (1989) "Blackboard Architectures and Applications" Academic Press.

[9] Ljunngberg, M., and Lucas, A., (1992) "The OASIS air traffic management system" Proc 2nd Pacific Rim Conf. On AI, Seoul, Korea.

[10] Jennings, N. R., (1995) "Controlling Co-operative problem solving in Industrial Multi-Agent Systems" Artificial Intelligence 75 (2) 195-240.

[11] Rao, A., Morley, D., Sekvestrel, M., and Murray, G. (1992) “Representation, selection and execution of team tactics in air combat modelling” Proc. 5th Australian AI Conf, 185-190.

[12] Ingrand, F. F., Georgeff, M. P., and Rao, A. S., (1992) “An architecture for real time reasoning and system control” IEEE Expert 7 (6).

[13] Burmeister, B., Haddadi, A., and Matylis, G., (1997) “Application of Multi-Agent Systems in Traffic and Transportation” IEE Proc. on Software Engineering 144 (1) 51-60.

[14] AAII (1994) “Distributed Multi-Agent Reasoning Systems (dMARS)” Technical Report, Australian Artificial Intelligence Institute, Melbourne, Australia.

[15] Lesser, V. R., and Erman, L., (1980) “Distributed Interpretation: A Model and Experiment”, IEEE Trans. On Computers, C-29(12):1144-1163.

[16] Lesser, V. R., and Corkill, D. D., (1983) “The distributed vehicle monitoring testbed: A Tool for Integrating Distributed Problem Solving Networks” AI Magazine, Fall, 15-33.

[17] Mason, C. L., (1995) “Co-operative seismic data interpretation for nuclear test ban treaty verification” Int. Journal of Applied AI, 9 (4) 371-400.

[18] Mason, C. and Johnson, R., (1989) “DATMS: a framework for distributed assumption based reasoning” in DAI Vol. II (eds. L. Gasser and M. Huhns) Pitman 293-318.