An Agent Architecture for Information Fusion and its Application to Robust Face Identification

Paul Robertson (paulr@ai.mit.edu)
Robert Laddaga (rladdaga@ai.mit.edu)

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Abstract

Information fusion is a difficult problem in general, and an especially difficult problem for vision systems. Evidence from multiple sources often cannot be combined in any straightforward manner, and significant reasoning and transformation operations may need to be performed on one piece of evidence before it can be usefully combined with another piece of evidence. Conventional approaches to information fusion are generally limited to cases where it is possible to combine signals at the signal processing level, before higher level interpretation takes place, or simply to process different signals independently, and then choose the "best" interpretation. Our approach combines evidence at multiple levels in the interpretation problem. We discuss the GRAVA architecture, and how it enables a flexible type of information fusion. The GRAVA architecture has been applied to several computer vision applications, and we use its application to face recognition to highlight the support for information fusion.

Keywords: Information Fusion, Computer Vision, Intelligent Agents, Model Based Reasoning.

1. Introduction

Information fusion has always been a difficult problem. Evidence from multiple sources cannot in general simply be combined. Standard approaches to information fusion either combine similar signals prior to interpretation, or process signals independently and then choose the best of the competing interpetations. An example of the former is a microphone array, which assumes that noise signals will be roughly the same at each individual microphone, and can be disgarded, while the voice signal will depend on the angle between the voice source and the individual microphone. This approach can work well with similar signals, but often is limited in the degree of improvement of signal interpretation. An example of the latter is a voting scheme for determining validity of results from parallel independent computations. This approach can also be very valuable, but is limited by inablility to apply disparate evidence sources to improve the performance of individual sensors or information sources. In both cases, there is often a combinatorial explosion due to the number of signals, or dependence between evidence sources.

One of the causes that prevents simple combination of evidence is when the evidence is from radically different types of information sources. For example, we may have aerial reconnaisance evidence of mobile missile launchers at a specific location in Iraq, but intelligence from human sources within the country that indicate the launchers are elsewhere. This type of problem is difficult because the "signals" can't be readily combined before interpretation, and after being interpreted, it is difficult to decide what relative weights to assign, and the degree of dependence in the evidence.

Vision systems have a built in difficulty with respect to information fusion: the visual signal is subject to interpretation at many different levels. Consider, for example, face recognition. If one camera delivers an image that can be interpreted as Jim's face at location x, and another camera delivers an image that indicates there is no face at the same location, it is not easy to see how best to combine such evidence. It is not easy to decide if it would be better to combine the raw camera images using some collection of operators, and then interpret the resulting combined image, or instead to weight the two pieces of evidence and choose the most likely, or to modify the interpretation of one image based on information from the other image. Image interpretation thus requires a more flexible approach to information fusion.

We can think of information fusion for vision as follows. In general, we want to form a hypothesis, find a way to apply evidence toward refuting or confirming the hypothesis, and form new hypotheses as needed. Information that refutes or confirms one hypothesis may also be used in the process of formulating a replacement hypothesis. However, rather than explicitly representing and reasoning about hypootheses, we use MDL to implicitly accomplish the same result. MDL allows us to combine evidence from disparate sources into multiple interpretations. Individual sources will contribute to lenthening or shortening the description that constitutes each individual interpretation. The selection of the interpretation with minimum description length has then implicitly fused the evidence from the multiple disparate sources.

In this paper, we first describe the general problem of face identification, then discuss the GRAVA architecture, and explain how it uses Minimum description length (MDL) to implement a more flexible information fusion approach.

2. The Face Recognition Problem

There are by now many face recognition systems that work well in constrained situations, but in natural environments, where lighting and pose can vary widely, they perform poorly. It is evident that multiple sources of information can yield a more robust face recognizer, if we can solve the problem of flexibly and efficiently combining the evidence from these sources.

For an image understanding system to interpret an image it is necessary for it to deal with the complexity in the image. Where we have been most successful in building vision systems has been in applications where the complexity and variation of the image content can be carefully managed. Examples of such situations include factory inspection applications and face recognition applications in which the face is deliberately positioned into a canonical location. Natural environments contain rich variety. It is very hard to build a single algorithm that can deal with the natural range of possibilities, but three aspects of the world provide a means of meeting the challenge: 1) any particular situation only has to deal with a subset of that variety at any time; 2) the subsets are not random but tend to occur in clusters or contexts; 3) we often have multiple sensors or other sources of information about the subject.

Most face identification and recognition systems work by measuring a small number of facial features given a canonical pose and matching them against a database of known faces. Frequently however in practical applications few frames show a full frontal face. Furthermore lighting may vary significantly. These factors frustrate attempts to identify a face. Many applications have much more relaxed recognition goals. If the task is to track people as they move throughout a monitored space the task may be to identify the individual from a relatively small set of people. For face profiles different models involving ear, eye, and nose may prove successful. By building a face recognizer that can fuse information from a collection of recognizer agents we can construct a recognizer that is robust to normal variations in the natural environment. This permits a much wider application of face recognition technology.

Our application involves recognizing people as they move about an intelligent space [4] in an unconstrained way. To better understand contexts consider the face "pose" contexts:

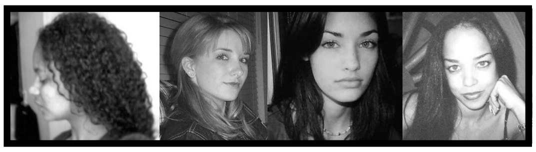

Figure 1. Four Pose Contexts

Figure 1 shows four pose contexts: "profile", "oblique", "off-center", and "frontal". The profile view is supported by agents that measure points along the profile of the face, the corner of the eye, and the lips. The oblique view with ear supports measurements of the ear and measurements of the position of the ear, eye, and nose. The triangle formed by the eye, ear, and nose help to determine the angle of the face to the camera which allows measurements to be normalized before recognition. The off-center view permits measurements of points on the eyes, nose, and mouth. The shape of the nose can be measured but the width of the base of the nose cannot be measured due to self-occlusion. The frontal view allows nose width to be measured but the nose shape cannot be measured. There are other contexts that include/exclude ears. The different contexts control, among other things, what models can be used for matching, what features can be detected and what transformations must be made to normalize the measurements prior to matching. This example shows contexts for pose but there are also contexts for lighting, race, gender, and age.

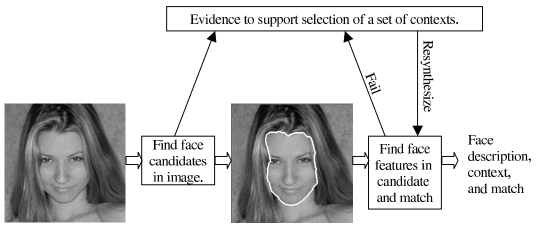
Evidence to support selection of a set of contexts.
Resynthesize
Fail
Find face candidates in image.
Find face features in candidate and match
Face description, context, and match

Figure 2. Recognizer Schematic

The recognizer supports a collection of face candidate finders, face models, feature finders, and normalization algorithms implemented as agents. The face recognition process is shown schematically in Figure 2. Face candidate finder agents look for face like shapes in the image and generate evidence that supports the selection of a set of contexts based on the shape and shading of the face candidate. Agents appropriate to the context are selected to make a special purpose face recognizer. If the recognizer doesn't succeed in finding appropriate features where they are expected to be the system self-adapts by using available evidence to select a more appropriate context, constructing a new recognizer, and trying again. The system iterates in this manner until appropriate lighting, race, age, gender, and pose contexts have been chosen and the best match has been achieved. Convergence on the right set of contexts is rapid because evidence in support of a context is collected each time an agent runs. The self-adaptive architecture responsible for the detection of contects and the selection of agents for a recognizer are discussed elsewhere [12].

In the following section we describe how evidence from the selected agents is fused to produce an interpretation.

3. Minimum Description Length

The Grounded Reflective Adaptive Vision Architecture (GRAVA) is a self adaptive architecture for image interpretation problems. GRAVA has already been successfully applied to satellite image interpretation [10, 11] and is now being used to identify faces in video images with unconstrained pose and lighting.

This section focuses on how MDL can be used in a novel way as a mechanism by which agents can cooperate and compete; how such an approach can provide a basis for communication between different semantic levels; how the resulting communication can facilitate information fusion; and to how this facilitates finding the best interpretation of the data. MDL then provides the means of finding the globally most likely interpretation.

Emphasis is given to models that can be learned from a corpus. One of the easiest ways of estimating probabilities—and hence description lengths—is to collect frequencies from representative data. The system described in the paper uses an image corpus for this purpose. Other mechanisms suggest themselves for estimating description lengths including using Qualitative Reasoning (QR) models and models of uncertainty.

3.1 MDL and Communication Theory

The notion of theoretical minimum code length described above provides a convenient way of restating the goal of "most probable interpretation" in communication theoretic terms.

An interpretation is a description that consists of many parts. Each part has an associated probability.

Consider the case of an image containing two blobs. If the probability of the first blob being a boy given by $P(blob_1 = boy) = P_{boy}$ and the probability of the second blob being a dog is given by $P(blob_2 = dog) = P_{dog}$ the probability that the image is one in which the first blob is a boy and the second blob is a dog is given by $P_{boy} * P_{dog}$ (from equation 3.2). An image containing n blobs such that for any $i < n$ the interpretation of $blob_i$ is given by $I_i(blob_i)$ and the probability of such an interpretation being correct is given by $P(I_i(blob_i))$ the goal is to maximize the probability (assuming conditional independence):

$$arg \max_{l_{1,n}} \prod_{i=1}^{n} P(I_i(blob_i)) \qquad (1)$$

In order to communicate the description using a theoretically optimal code would require a description length of: $(-log_2(P_{boy}))+(-log_2(P_{dog}))$. For an image containing n blobs as defined above the goal is to choose interpretations $I_1...I_n$ so as to minimize the description length:

$$arg \min_{l_{1,n}} \sum_{i=1}^{n} -log_2(P(I_i(blob_i))) \qquad (2)$$

Finding the most probable interpretation is identical to finding the minimum description length (MDL).

The approach raises two significant issues.

1. How to estimate P.

2. How to find the global MDL.

Neither of these issues is straightforward. In this paper we depend upon estimating P from frequencies obtained from a corpus. For a general mechanism for finding MDL we employ a Monte Carlo sampling scheme.

3.2 Monte-Carlo methods

The MDL Agent architecture described in this paper addresses the need to integrate knowledge at different semantic levels. To understand an image that consists of high level components such as objects and of low level features such as lines and textures we need to integrate different levels of processing.

Single thread of control solutions, such as the blackboard and forward chaining approaches, depend upon taking a path towards a solution and backtracking past failures until a solution is found. The same is true of subsumption. These are essentially depth first search for a solution—not search for the *best* solution. In a robot navigation problem, we may be happy if the robot negotiates the obstacles in the environment and finally ends up at the destination. In interpretation problems, just finding a solution is not good enough. An English sentence can have many plausible parses. Most of them do not make sense. ie: syntactically okay, but semantically garbage.

Markov Chain/Monte Carlo methods (MCMC) have become popular recently in computer vision[5, 7] but have been limited to modeling low level phenomenon such as textures [6]. In natural language understanding, use of hidden Markov models has been successful as an optimization technique with certain restricted kinds of grammar. Problems that can be described as Hidden Markov Models (HMM) [1] can yield efficient algorithms. For example, in natural language understanding, some grammars permit efficient algorithms for finding the most probable parse. Stochastic Context Free Grammars (SCFGs) can be parsed so as to find the most probable parse in cubic time using the Viterbi algorithm [14]. Only the simplest grammars and problems can be solved efficiently in this way, however, and for the more interesting grammars and for more complex problems in general, other techniques must be used. Certainly something as loosely defined as an agent system incorporating semantics from multiple levels would rarely fit into the HMM straitjacket.

Even for natural language processing, finding the best solution can be prohibitively expensive when the Viterbi algorithm can't be employed. In visual processing, with images whose complexity greatly exceeds that of sentences, and which are three dimensional [as opposed to the linear arrangement of words in a sentence], finding the best solution is infeasible. Approximate methods are therefore attractive. Monte-Carlo techniques are very attractive in these situations.

In an ambiguous situation, such as parsing a sentence, in which many [perhaps thousands] of legal parses exist, the problem is to find the parse that is the most probable. If the problem can be defined in such a way that parses are produced at random and the probability of producing a given parse is proportional to the probability that the parse would result from a correct interpretation, the problem of finding the most probable parse can be solved by sampling. If P is a random variable for a parse, the probability distribution function (PDF) for P can be estimated by sampling many parses drawn at random. If sufficiently many samples are taken the most probable parse emerges as the parse that occurs the most frequently. Monte Carlo techniques use sampling to estimate PDF's. In this paper we generalize the idea of parsing in order to build an agent system.

Monte Carlo methods are attractive for a number of reasons:

1. They provide an approximate solution to search of search spaces whose combinatorics are prohibitive.
2. By adjusting the number of samples, the solution can be computed to an arbitrary accuracy.
3. Whereas the best solution can not be guaranteed by sampling methods, measuring standard error during the sampling phase allows the number of samples to be adjusted to yield a desired level of accuracy.

In this paper, we are primarily interested in composing systems of agents in such a way that the most probable global descriptions are produced. This brings us directly to the issue of how to select among agents that may be applicable at a given place in a computation. We have considered three approaches in the foregoing:

1. The selection is carefully programmed. Hearsay II did this with its focus of control database.
2. Local competition among agents.
3. Monte Carlo sampling to estimate global minima.

3.3 Agent Selection Paradigms

Autonomous agents are expected to operate without the intervention of a central control mechanism [such as a focus of control database]. One approach to the agent selection problem that has been the focus of considerable attention, is the notion of a market based approach. The idea is that an agent wishes to farm out a subtask to another agent capable of performing the subtask. Agents that are candidates to perform the subtask compete by bidding a price. This often works well, producing efficient solutions. However, two problems arise in such systems:

1. Selecting an appropriate basis for cost computations so that the bidding is fair.
2. Because the bidding is piecewise local, such systems are prone to find local minima and miss the global minima.

Our approach addresses these two problems as follows.

The basis for cost computation is description length. Description length is the correct measurement in an interpretation problem because it captures the notion of likelihood directly: $DL = -log_2(P)$.

Monte Carlo sampling allows us to avoid the problem of finding unwanted local minima.

In the following section, we describe the architectural objects that implement these ideas in the GRAVA architecture. In the third section we highlight the idea that Monte Carlo sampling can solve problems similar to those for which systems of inhibition have been used by demonstrating the agent architecture on a reading problem that was previously solved using a multi-layer inhibition based neural network with downward control by McClelland [9]. Our approach, however, achieves highly robust results without the use of any downward flow of control.

4. MDL Agent Architecture

The architecture is built from a small number of objects: Models; Agents; Interpreters; Reflective Levels; and Descriptions.

All of these terms are commonly used in the literature to mean a wide range of things. In the GRAVA architecture they have very specific meanings. Below, we describe what these objects are and how they cooperate to solve an interpretation problem.

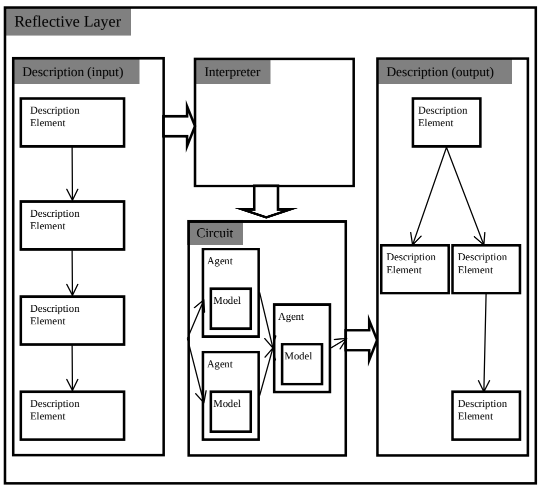

Figure 3. Objects in the GRAVA Architecture

Figure 3 shows the objects that make up the architecture. A reflective layer takes an input description Δ_{in} and produces an output description Δ_{out} as its result. A description consists of a collection of description elements $< \epsilon_1, \epsilon_2, ..., \epsilon_n >$. The output description is an interpretation ($I \in Q(\Delta_{in})$) of the input where $Q(x)$ is the set of all possible interpretations of x.

$$\Delta_{out} = I(\Delta_{in}) \tag{3}$$

The goal of a layer is to find the best interpretation I_{best} which is defined as the interpretation that minimizes the global description length.

$$arg \min_{I_{best}} DL(I_{best}(\Delta_{in})) \tag{4}$$

The interpretation function of the layer consists of an interpretation driver and a collection of connected agents. The interpretation driver deals with the formatting peculiarities of the input description (the input description may be an array of pixels or a symbolic description). The program is made from a collection of agents wired together. The program defines how the input will be interpreted. The job of the interpreter/program is to find the most probable interpretation of the input description and to produce an output description that represents that interpretation.

The GRAVA architecture allows for multiple layers to exist in a program and there are [reflective] links between the layers.

Below, we describe in greater detail the purpose, protocol, and implementation of the objects depicted in Figure 3. We maintain a dual thread in the following. On the one hand, we describe the GRAVA architecture abstractly, on the other, we also describe the actual implementation that we have developed.

4.0.1 Description

A description Δ consists of a set of description elements ϵ.

$$\Delta = < \epsilon_1, \epsilon_2, ..., \epsilon_n > \tag{5}$$

Agents produce descriptions that consist of a number of descriptive elements. The descriptive elements provide access to the model, parameters, and the description length of the descriptive element. For example, a description element for a face might include a deformable face model and a list of parameters that deform the model face so that it fits the face in the image. A description element is a model/parameters pair.

The description length must be computed before the element is attached to the description because the agent must compete on the basis of description length to have the descriptive element included. It makes sense therefore to cache the description length in the descriptive element.

The description class implements the iterator:

```
(for Description|des fcn)
```

This applies the function "fcn" to every element of the structural description, and this enables the architecture to compute the global description length:

$$DL(\Delta_{out}) = \sum_{i=1}^{n} DL(\epsilon_i) \tag{6}$$

4.0.2 DescriptionElements

Description elements are produced by *agents* that *fit models* to the input.

Description elements may be implemented in any way that is convenient or natural for the problem domain. However the following protocol must be implemented for the elements of the description:

```
(agent <Element>)
```

Returns the agent that fitted the model to the input.

```
(model <Element>)
```

Returns the model object that the element represents.

```
(parameters <Element>)
```

Returns the parameter block that parameterizes the model.

```
(descriptionLength <Element>)
```

Returns the description length in bits of the description element.

Implementations of description elements must inherit the class DescriptionElement and implement the methods "agent", "model", "parameters", and "descriptionLength".

For readability we print description elements as a list:

```
(<model name> . <parameter list>)
```

4.0.3 Models

Fitting a model to the input can involve a direct match but usually involves a set of parameters.

Consider as input, the string:

```
‘‘t h r e e  b l i n d  m i c e’’
```

We can interpret the string as words. In order to do so, the interpreter must apply word models to the input in order to produce the description. If we have word models for "three", "blind", and "mice" the interpreter can use those models to produce the output description:

```
((three) (blind) (mice))
```

The models are parameterless in this example. Alternatively we could have had a model called "word" that is parameterized by the word in question:

```
((word three) (word blind) (word mice))
```

In the first case there is one model for each word. In the case of "three" there is an agent that contains code that looks for "t", "h", "r", "e", and "e" and returns the description element "(three)". In the second case there is one model for words that is parameterized by the actual word. The agent may have a database of words and try to match the input to words in its database.

Consider the two examples above. If the probability of finding a word is 0.9 and the probability of the word being "three" is 0.001 the code length of "(word three)" is given by:

$$\begin{aligned} DL(wordthree) &= DL(word) + DL(three) \\ &= -log_2(p(word)) - log_2(p(three)) \\ &= -log_2(0.9) - log_2(0.001) \\ &= 0.1520 + 9.9658 \\ &= 10.1178 bits \end{aligned} \tag{7}$$

The second approach, in which a separate agent identifies individual words would produce a description like "(three)". The model is "three" and there are no parameters. The likelihood of "three" occurring is 0.001 so the description length is given by:

$$\begin{aligned} DL(three) &= -log_2(p(three)) \\ &= -log_2(0.9 * 0.001) \\ &= 10.1178 bits \end{aligned} \tag{8}$$

That is, the choice of parameterized vs. unparameterized doesn't affect the description length. Description lengths are governed by the probabilities of the problem domain. This allows description lengths produced by different agents to be compared as long as they make good estimates of description length.

For a more realistic example, consider the case of a principle component analysis (PCA) model of a face [2]. A PCA face model is produced as follows. First a number n of key points on a face are identified as are their occurrences on all of the images. The shape of the face ψ_i is defined by a vector containing the n points. A mean shape is produced by finding the average position of each point from a set of example face shapes.

$$\bar{\psi} = \frac{1}{n}\sum_{i=1}^{n}\psi_i \tag{9}$$

The difference of each face from the mean face $\bar{\psi}$ is given by:

$$\delta\psi_i = \psi_i - \bar{\psi} \tag{10}$$

The covariance matrix S then is given by:

$$S = \sum_{i=1}^{n}\delta\psi_i\delta\psi_i^T \tag{11}$$

The eigenvectors p_k and the corresponding eigenvalues λ_k of the covariance matrix S are calculated. The eigenvectors are sorted in descending order of their eigenvalues. If there are N images, the number of eigenvectors to explain the totality of nN points is N, typically large. However, much of the variation is due to noise, so that $p << N$ eigenvectors suffices to account for (say) 95% of the variance. The most significant of the eigenvector-eigenvalue pairs are selected as the principal components.

The resulting face model consists of a mean face shape $\bar{\psi}$ and a set of eigenvectors and weights such that any face shape ψ_p can be approximated by:

$$\psi_p = \bar{\psi} + \mathbf{Pb}, \tag{12}$$

where **P** is the vector of eigenvectors and **b** is the vector of weights. The weights are a measure of how much the model must be distorted in order to match the face ψ_p.

The above formulation of a face shape model describes a parameterized model. The weights are the parameters and the mean shape and vector of eigenvectors is the model. Algorithms exist for fitting such shape models to data. These algorithms first identify a key component and then, using the mean shape model, search for the other features (often edges) near the place where the mean suggests it should be. When the feature is found, its actual location is used to define a distance from the mean. This is repeated for feature points in the model. A set of weights is calculated which represents the parameterization of the model.

4.0.4 Agents

The primary purpose of an agent is to fit a model to its input and produce a description element that captures the model and any parameterization of the model.

We implemented the atomic computational elements in GRAVA as agents. The system manipulates agents and builds programs from them but does not go beneath the level of the agent itself. The agent allows conventional image processing primitives to be included in the GRAVA application simply by providing the GRAVA agent protocol. We might have used methods if we were building a language rather than an architecture. GRAVA agents are not autonomous agents. They depend upon other agents to reason about them and to connect them together to make programs.

An agent is a computational unit that has the following properties:

1. It contains code which is the implementation of an algorithm that fits its model to the input in order to produce its output description.

2. It contains one or more models [explicitly or implicitly] that it attempts to fit to the input.

3. It contains support for a variety of services required of agents such as the ability to estimate description lengths for the descriptions that it produces.

An agent is implemented in GRAVA as the class "Agent". New agents are defined by subclassing "Agent". Runtime agents are instances of the appropriate Agent class. Generally Agents are instantiated with one or more models. In our system all models are learned from the corpus.

The protocol for agents includes the method "fit" that invokes the agent's model fitting algorithm to attempt to fit one or more of its models to the current data.

```
(fit anAgent data)
```

The "fit" method returns a (possibly null) list of description elements that the agent has managed to fit to the data. The interpreter may apply many agents to the same data. The list of possible model fits from all applicable agents is concatenated to produce the candidate list from which a Monte Carlo selection is performed.

4.0.5 Interpreters

An *interpreter* is a *program* that applies *agents* in order to produce a structural description output from a structural description input.

A scene interpretation program may include agents for face recognition—such as the PCA face shape agent described above—and may include other agents that recognize other things that would be found in an image such as trees, buildings, and roads. The interpreter could be hand-assembled or it could be generated.

4.0.6 Monte Carlo Agent Selection

A recurring issue in multi-agent systems is the basis for cooperation among the agents. Some systems assume benevolent agents where an agent will always help if it can. Some systems implement selfish agents that only help if there is something in it for them. In some cases the selfish cooperation is quantified with a pseudo market system.

Our approach to agent cooperation involves having agents compete to assert their interpretation. If one agent produces a description that allows another agent to further reduce the description length so that the global description length is minimized, the agents *appear* to have cooperated. Locally, the agents compete to reduce the description length of the image description. The algorithm used to resolve agent conflicts guarantees convergence towards a global MDL thus ensuring that agent cooperation "emerges" from agent competition. The MDL approach guarantees convergence towards the most probable interpretation.

When all applicable agents have been applied to the input data the resulting lists of candidate description elements is concatenated to produce the candidate list.

The *monteCarloSelect* method chooses one description element at random from the candidate list. The random selection is weighted to the probability of the description element.

$$P_{elem} = 2^{-DL(elem)} \tag{13}$$

So, for example, if among the candidates, one has a description length of 1 bit and one has a description length of two bits, the probabilities of those description lengths is 0.5 and 0.25 respectively. The *monteCarloSelect* method would select the one bit description twice as often as the two bit description.

5. Conclusion

The architecture described is based on two ideas:

1. That for interpretation problems global MDL is the goal;

2. That global MDL can be approximated by Monte-Carlo sampling.

Because MDL is not specific to any particular problem it provides a common currency or "gold standard" for use in market model agent systems. Because the architecture is specialized to solving interpretation problems there are problems for which it is not appropriate. Nevertheless a great many interesting problems can be cast in the form of interpretation problems.

Because the foundations upon which the architecture is built are well understood the behavior and performance of the architecture can yield to some level of analysis—such as convergence analysis. The architecture has some interesting characteristics:

1. Cooperation between agents at different semantic levels is an emergent property of the architecture as was demonstrated by the "reading" example given in this paper.

2. Robustness within the limits of the programs domain is realized by virtue of measuring how local choices affect global description length.

3. There is an implicit information fusion model.

Most attempts at reasoning about uncertainty [13, 8, 3, 15] attempt to bring together contributions so as to make a local decision. When local evidence is sufficient these methods work well but when the sources of evidence become less straightforward the approaches get bogged down. Numerous problems occur—most notably that required probabilities are often not available and that in order for the approaches to be tractable it is necessary to assume conditional independence. In any complex system the practical issues are immense.

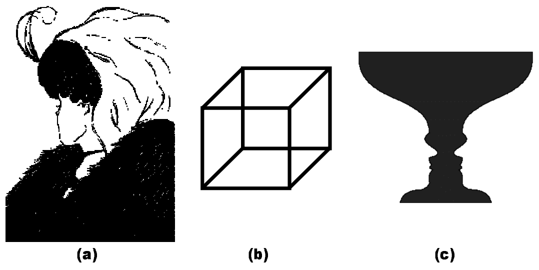

Figure 4. Ambiguous Visual Interpretations

Figure 4 is an example of three standard illusions where there are two almost equally plausible interpretations. The resolution of such ambiguities (where possible) is at the heart of the information fusion problem. For example, the image Figure 4(a) may initially appear to be a view from the left rear of the head of a young woman but the features have been cunningly arranged by the artist so that we can easily pick an alternative interpretation—that of an old woman. In Figure 4(a) the ear of the young woman is interpreted as an eye when the image as a whole is seen as an old woman.

Imagine a program that attempts to combine evidence to interpret the feature that is the young girl's ear in Figure 4(a). The information fusion problem must consider what interpretation model is being used and what relationship that has with feature models being considered for the "ear". Locally the interpretation of the "ear" feature as an "eye" or as an "ear" may be quite similar. The choice however effects the interpretation of surrounding features (nose and chin for example). By trying to treat this as a local information fusion problem conventional methods of information fusion potentially demand that the entire structural complexity of the problem be considered at each local decision point.

The implicit information fusion model in the GRAVA architecture depends upon the effect that local decisions have upon the global interpretation. In the reading example described in this paper information from four different kinds of sensor were utilized in providing evidence in support of interpretations as characters. A less probable interpretation of a feature will be selected if it gives rise to a shorter global description.

References

[1] L.E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a markov process. *Inequalities*, 3:1–8, 1972.

[2] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. In H. Burkhardt and B. Neumann, editors, *Proceedings, European Conference on Computer Vision 1998*, volume 2. Springer, 1998. pages 484-498.

[3] A.P. Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society, Series B*, 30:205–247, 1968.

[4] R. A. Brooks et al. The intelligent room project. In *Proceedings of the Second International Cognitive Technology Conference (CT'97), Aizu, Japan*, 1997.

[5] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[6] N. Karssemeijer. Stochastic model for automated detection of calcifications in digital mammograms. *Image and Vision Computing*, 10/6:369–375, 1992.

[7] T.W.E. Lau and Y.C. Ho. Universal alignment probabilities and subset selection for ordinal optimization. *Journal of Optimization Theory and Applications*, 93:455–489, 1997.

[8] W.B. Mann and T.O. Binford. Probabilities for bayesian networks. In *Proceedings Image Understanding Workshop*. Morgan Kaufman, San Francisco., 1994.

[9] J.L. McClelland and D.E. Rumelhart. The programmable blackboard model of reading. In *Parallel distributed processing.*, pages 122–169. MA: MIT Press, 1986.

[10] P. Robertson. A corpus based approach to the interpretation of aerial images. In *Proceedings IEE IPA99*. IEE, 1999. Manchester.

[11] P. Robertson. *A Self-Adaptive Architecture for Image Understanding*. PhD thesis, University of Oxford, 2001.

[12] P. Robertson and R. Laddaga. A self-adaptive architecture and its application to robust face identification. In *Pacific Rim Conference on Artificial Intelligence 2002*. Springer-Verlag, 2002.

[13] G. Shafer. *A Mathematiccal Theory of Evidence*. Prinston University Press, 1976.

[14] A.J. Viterbi. Error bounds for convolution codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[15] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 100:9–34, 1999.