Model - based vision

A.W.M. Smeulders *University of Amsterdam, Amsterdam, The Netherlands*
J.S. Duncan *Yale University Medical School, New Haven, Connecticut, USA*

Abstract

It is argued that the inclusion of model knowledge is important in the advancement of image analysis software, to separate the specialities of the current problem area from general methods of image analysis. In the these two presentations an overview is given of two classes of image analysis solutions: symbolic model-driven image analysis and physical model-driven analysis. For symbolic models a system architecture consisting of detectors (through which the access is to the various representations of the data) with encapsulated knowledge on the signs to detect, a reasoning module to propagate the uncertainty. Symbolic model-driven analysis is evaluated on an extended example: map interpretation. Physical models are presented when an integral view on the data is needed. The extended example in this case is: heart wall motion estimation. Whereas the lectures discussed the more general aspects of symbolic- and model-driven analysis, this text is confined to the extended examples.

Keywords: model-driven image analysis, symbolic, physical, system design, map interpretation, cardiac image analysis

1 Introduction

The state of the art in vision research is far from formulating all purpose solutions. In fact, a vision system is ill-posed when in the specification a description of the domain, the purpose of the system or the criterion for success are missing.

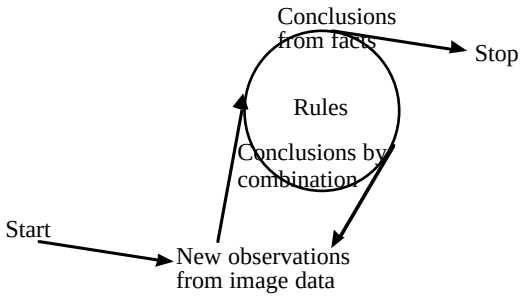
In this contribution, we concentrate on the inclusion of domain specific knowledge in vision solutions for the purpose of segmenting the image and recognising the object. Given a digital image as a data field and a generic model of the object can we come to a recognition of the object represented by the data? In this presentation, we discuss two (very) different types of models, theoretically and by extended example: the force based models and the rule based models.

In the force driven domains, the object may assume a continuum of different forms each somewhat different. The strategy leading to its recognition and description of its state is to formulate a geometric model of the object. The geometric model is steered by one or more parameters. By parameter adjustment, the generic model is instantiated onto the specific view of the specific object as encountered by the data field configuration. In the space of all admissible object shapes, the (iterative) search is for the one best matching the data field. Not every shape is plausible. Domain knowledge is implemented as constraints on the shape (smoothness, cornedness) as well as in the criterion where to best match (at places of a steep outward pointing intensity gradient).

Force driven models are frequently integral models, as they assume one uniform mechanism which underlies the shape as a whole. Model - based image analysis of this kind results in a parameterized representation of the object. In section 2, we present an example of heart wall motion detection.

Symbolic domain models are based on an entirely different way to integrate domain knowledge in the analysis. Now, it is assumed that domain knowledge is the consequence of a formalised verbal description: in formalised verbal statements. As an example of

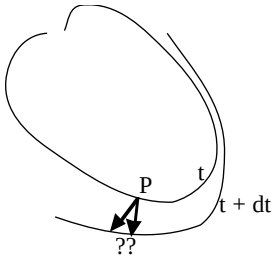
knowledge representation consider the statement: "If (such) is almost certainly present in the image near (this), then, quite likely, the image holds (that)." The first-level rules determine what observations can be made on the image data field. The outcome is a fact on the presence or absence of visual entities, and passed on to the next level. Higher level rules in the reasoning engine lead to a conclusion about the facts known thus far or to the quest for new information from the data field. The iteration of conclusion and new fact search or conclusions results in the reasoning loop:



We will present an example on this AI - style model driven analysis in section 3 on paper map interpretation.

2 Force - driven vision

The force model - driven vision is exemplified on a difficult problem: the segmentation and motion estimation of the heart wall in 3D - recordings. The problem is known as the 3D non - rigid motion estimation, where the problem is to track points on the heart wall over time. Imagine the heart wall moves from state at time t to the shape at time t+dt. Then the problem arises where does the point P go at t+dt. Where it is certain that P remains on the expanding surface, it is the question precisely where does it go?



The approach to non-rigid motion tracking problem is based on the 3D-bending model. We follow [1] very closely.

The first step is to find in the image data all points which constitute the border of the heart by searching for the best overall match of the deformable model to the data field. We discuss the two dimensional case. A model is a string of (x, y) - co-ordinates, and denoted by **x**(.). The precise shape the model **x**(t, **p**) is in depends on t, the displacement along the contour, and on parameter vector **p** used to bend the model in a large variety of different forms.

The match quality $Q_1(t)$ to match the model to the data averaged over all positions t is said to be made up of two terms [2]. One to make sure the model connects the places where the maximum outward pointing gradient is in the data:

$$Q_1(t) = \int_t \nabla f(\mathbf{x}(t, \mathbf{p})) \,.\, \mathbf{x}'_{\perp}(t, \mathbf{p}) \; dt$$

where the local gradient $\nabla f(\mathbf{x})$ is computed from the image data $f(\mathbf{x})$ by differential filters, and $\mathbf{x}'_{\perp}(t, \mathbf{p})$ is the derivative of the normal to the curve. The other one takes care that the contour is smooth:

$$Q_2(t) = \int_t \kappa'(\mathbf{x}(t, \mathbf{p}))^2 \, \| \mathbf{x}'(t, \mathbf{p}) \| \, dt \quad . \quad \int_t \| \mathbf{x}'(t, \mathbf{p}) \| \, dt$$

where the local curvature change κ' is computed from the contour model, and $\mathbf{x}'(t, \mathbf{p})$ is the derivative of the tangent to the curve. The best integral match is reached by optimising the parameters $\mathbf{p}$ of the path $\mathbf{x}$ generated by the model $\mathbf{x}(t, \mathbf{p})$ is such that the integral quality matches best. The optimisation procedure is:

$$\mathbf{x}^* = \arg\max_{\mathbf{p}} Q_1(\mathbf{p}) + \lambda \, Q_2(\mathbf{p})$$

where λ takes care for a problem dependent balance between the two quality factors. This procedure yields a model-based segmentation of the heart data, slice by slice. The segmentation result of each of these slices combined gives a segmentation of the heart volume.

To model the motion of the 3D heart, a wire triangulation is matched to the segmentation result. For each patch, fit a biquadratic surface to compute the differential characteristics of the surface point. The biquadratic fit involves finding in each triangular patch the six coefficients:

$$a\,x^2 + b\,xy + c\,y^2 + d\,x + e\,y + f$$

where the co-ordinates (x,y) are measured along the tangent directions, and the z - co-ordinate is measured along the normal direction. In matrix form, the coefficient vector which minimises the fit error is:

$$(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{Z}$$

where $\mathbf{Z}$ is a column vector of the $\mathbf{z}_i$ values. $\mathbf{A}$ is a n x 6 matrix, with n is the number of data points. Row i consists of $[x_i^2, x_i y_i, y_i^2, x_i, y_i, 1]$. The result of this procedure is the computation over triangular patch $\mathbf{s}$ of a value $K_0(\mathbf{s})$ denoting local curvature. Because curvature information is rather unstable, the curvature field is smoothed by a similar procedure as before:

$$K^*(s) = \arg\min \int_{\mathbf{S}} C_d(\mathbf{s}) \, (\, (K(\mathbf{s}) - K_0(\mathbf{s}) \,)^2 + \frac{\partial K}{\partial \mathbf{s}} \,) \, d\mathbf{s} \; .$$

S is the surface domain, and C_d is a confidence measure derived from the matrix estimation expressing goodness of fit, and hence reliability for that patch. In K*(**s**), we have obtained a plausible estimation of the curvature over the heart surface.

The physical model of the heart wall is a thin-plate loosely constrained to deform in a predetermined way. The potential energy of an ideal thin flexible flat plate of elastic material which is a measure of the strain energy of the deformation is given:

$$e = \frac{\kappa_1^2 + \kappa_2^2}{2}$$

where κ_1 and κ_2 are the two principal curvatures of the surface. This measure is invariant to 3D - rotation and translation. The above physical quantity is slightly modified to define the energy required to bend a curved plate in a shape at time t, to a newly deformed state at t+Δt as:

e_d = (**Error!**- **Error!**)

This equation assumes that the corresponding points on the surfaces are known. Under the assumption that surface deforms only slightly and locally within a small time interval, for each sampled point **P** on the surface at time t, a local area is searched on the surface at t+Δt. The point within the search window on the second surface that is best matched (i.e. minimising the bending energy deformation) is chosen as the point $\mathbf{P}_{t+\Delta t}$ corresponding to $\mathbf{P}_t$. The value of bending deformation energy is used as an indicator of the uniqueness of the match. The matching process yields a set of shape-based motion vectors $\mathbf{D}_0(\mathbf{s})$, pointing from any point $\mathbf{P}_t$ of the one surface to $\mathbf{P}_{t+\Delta t}$ on the next one in the sequence.

Again, $\mathbf{D}_0(\mathbf{s})$ is a rough vector field, so again smoothing is necessary by the physically plausible constraint of locality. Strong, unique matches from the bending energy matching process should be preserved, while ambiguous matches should be smoothed over by their neighbouring matches. The confidence measures from the initial match are used as weighting coefficients in the smoothing functional. The functional for smooth motion estimation is similar to the curvature smoothing functional:

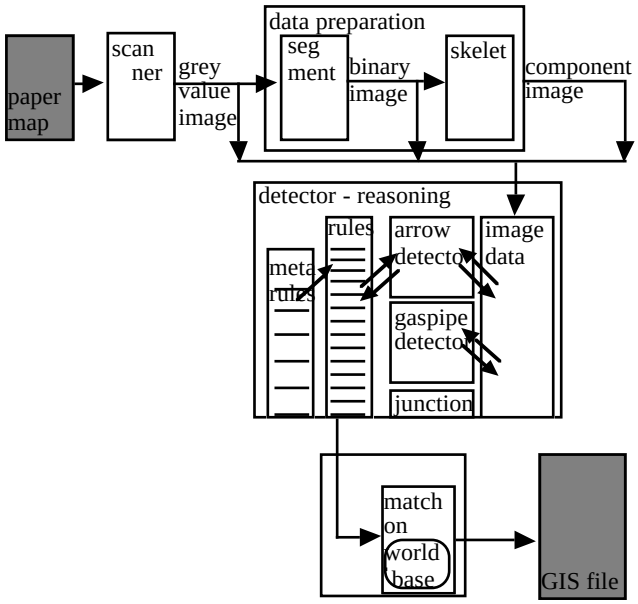
$$\mathbf{D}^*(\mathbf{s}) = \arg\min \int_{\mathbf{S}} C_d(\mathbf{s}) \left(\mathbf{D}(\mathbf{s}) - \mathbf{D}_0(\mathbf{s})\right)^2 + \frac{\partial \mathbf{D}}{\partial \mathbf{s}} d\mathbf{s} \ .$$

At the end of this process we have yielded a reliable estimate for motion estimation of the heart surface in vector field **D***(**s**).

3 AI model - driven vision

We now turn to the case where the knowledge of the domain has a predominantly rule - based character. Such a characteristic will frequently be encountered when there is a deliberate intention behind the object of which the image is to be analysed. A clear example is an engineering drawing, as 1) they are meant to be understood in one and precisely one way, 2) they imply a specific set of actions during the drawing, and 3) the imply a sequence of specific actions in its understanding. These actions can be specified in symbolic statements, in principle, as well as its order. So, the contents of a drawing can be known, in principle.

To demonstrate the principle of symbolic model - driven image understanding, consider the problem of a paper map to be turned into an electronic file by image analysis. The process will consist of the following steps: scanning -- ink detection -- the reasoning loop -- match to real world -- store. We go over the steps one by one.



The paper map will be scanned by feeding it through a large size scanner. We require such a scanning focus, transport stability and resolution that we have enough spatial resolution to be able to detect the presence and location of graphical symbols in spite of the presence of noise, stains and repairs of the map. Grey value scanning is a safeguard against jagged line segments as well as line rupture when the line intensity locally fades. As a function of line width and effective grey value resolution, 400 dots per inch 8 bit grey value is used. A digital image of a map typically measures more than 10,000 x 10,000 pixels, or 100 Mbyte of data. See the example in Figure 3.

The next step, ink detection, is introduced to bootstrap the reasoning process as well as speeding up the analysis process later on. Target here is to discriminate all instances of ink in the field of view from the dirt, spills and other irregularities. To that end, locally adjusted segmentation schemes build on knowledge of the minimum and maximum admissible pencil width profile. Also in use is a skeleton algorithm, indicating the heart of each line, at crossings taking into account knowledge that two pencil movements have created a locally increased line width. Output of this step is a modified grey - values image where the contamination's have been removed, a binary file indicating the locations where the map contains ink (if all is well), and a line-segment file introduced for efficiency only listing all stretches of uninterrupted line strokes. These three files are handed over to the reasoning - loop.

The purpose of the reasoning loop is the propagation of uncertainty from observations on the scan data to a conclusion about the map structure consisted with the rules of the map maker. At closer look, it consists of meta - reasoning, reasoning and detection.

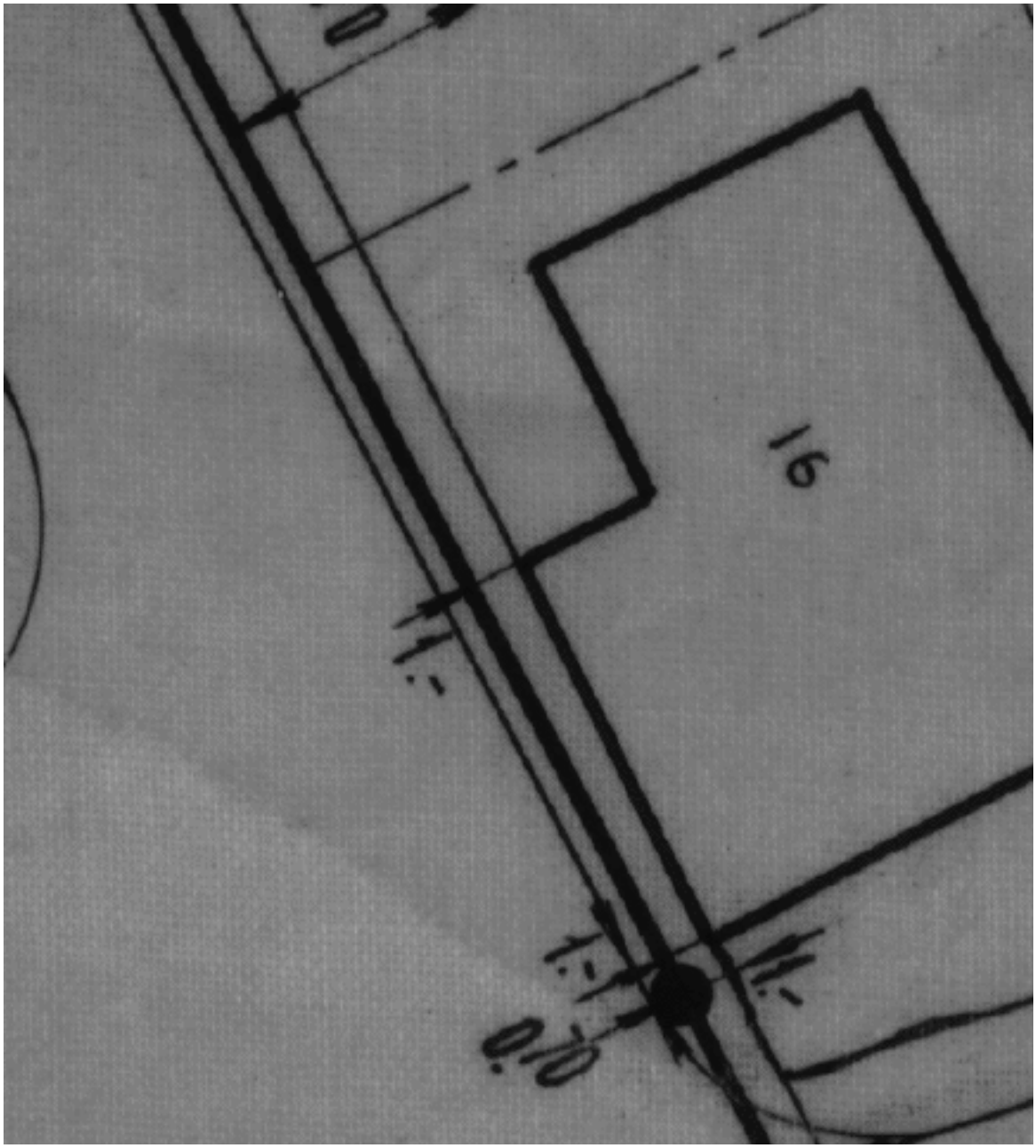


Figure 3: Example of a paper utility map scan, only a very small part is shown. Data kindly provided by the PNEM through the TopSpin contract.

Graphic sign recognition occurs through a set of computational routines, called detectors, one to each type of sign. The list of all signs is known as the legend, but in every day practice, the rules derived from the legend are to be completed with regional differences in style, implicit rules and abstract and composed symbols.

In the example, the graphic sign of a "lit" always is indicated by a solid circle with the line indicating the gas running through its middle. Such a symbol has a fixed ideal appearance, with only the position of the gas line and its diameter as parameters. This is the most simple case. The graphical sign of houses is characterised by a fixed line width in an open or closed poly-line. No fixed shape is present here, and the sign detector is to be composed of a line width detector, followed by a straight line segment detector

working on the result of the previous one, followed by a house detector building the straight line segments into a house, according to the admissible rules. Least rigid in rules and hence shape are the dashed lines linking the contents of the gas line to the drawing of the line.

The detector will contain knowledge of the dashing, the line width, repetition patterns, starting position, etc. All in all 20 different graphical sign detecting algorithms are needed for the example, each of which may contain some knowledge of the map maker's rules. Generally, for practical applications any detector will be characterised by non-perfect detection. Specifically it will erroneously assign an object to its target (a false positive finding), or it will left unrecognised an object which in fact should have been detected (false negative). Any parameterized detector can be tuned on a Receiver Operator Characteristic curve, better known as ROC-curve, to a certain false positive value from which its false negative rate will follow, or vice versa. The tuning cannot improve both error rates unless the quality of the detector is improved. Where to tune a detector depends on the application. If there are very few signs and they are very crucial to observe, a low false negative is best at the expense of a poor false positive rate, creating quite a few false alarms.

Once the graphic signs have provided a clue, reasoning may start to explain the instantiations of the signs according to the map makers rules. The lowest layer of rules will absorb input from the detection modules and produce a conclusion at a higher level for the map at hand. This then feeds into a higher level rule which leads to a high level conclusion, until all observed signs are considered, none left unused and the highest level conclusion has been reached that the map interpretation is consistent.

The reasoning engine will perform actions of the following types: detected signs (facts) feed into a rule (knowledge) leads to a consisted conclusion. Note that a conclusion once it has been reached has the status of fact for the map at hand. Inconsistency occurs when two conclusions demonstrate a conflicting outcome or when observations are made for which there is no rule to explain them. In the first case, observations should be reconsidered or, alternatively, the rules of the model may be wrong. In the second case, again observations may be faulty, or the model may be incomplete.

In a reasoning engine facts are connected with conclusions by rules, known to describe the domain. There are several candidates for such a reasoning. In most practical cases, strict logic is insufficiently robust to deal with situations where detection may be faulty. In a strict logic reasoning engine, rules are of the type:

Rule i: ('fact') implies 'conclusion' is true.

where 'fact' can be composed of more than one facts by '.and.', '.or.' and '.not.' operators, and 'conclusion' may be used as fact in other rules. Note that if two rules point at the same 'conclusion' and they are both used in one map, they both should lead to the same logical 'conclusion', otherwise the system is inconsistent, i.e. the interpretation of the map runs into a dead end.

With strict logic reasoning, any false detection leads to inconsistency or to a consistent but erroneous conclusion. Therefore, propagating some form of uncertainty may be helpful, especially when the rules of the domain contain redundancy. In the example, a doubly arrowed line is always drawn between a house and a gas line, so the uncertain detection of a one, possibly two, arrow-headed line the surrounding graphic symbols may help out deciding. In the Mycin - formalism, the central paradigm is that compatible certainties are attached to facts as well as to implications, in spite of the fact that the one certainty is observational and the other one included in the model knowledge. By making them equal they can be manipulated as follows. Uncertainty is indicated by:

$u = +1$	most certainly true
$u = 0$	indifferent
$u = -1$	certainly not true

The combinatorics of Mycin are defined as:

.and.	$fact_1$ .and. $fact_2$	$u_r = \min\{ u_1, u_2 \}$
.or.	$fact_1$ .or. $fact_2$	$u_r = \max\{ u_1 , u_2 \}$
.not.	.not. fact	$u_r = - u_1$
implies ('fact') implies 'conclusion'		$u_r = u_{fact} . u_{imply}$

There is a special combinatoric rule for combining two different rules pointing at *same* conclusion. Rule 1 has a resulting certainty of u_{r1}., and rule 2 has a resulting certainty of u_{r2}. Then,

$u_{combined} = u_{r1} - u_{r1}.u_{r2} + u_{r2}$	for $u_{r1} > 0$ and $u_{r1} > 0$,
$u_{combined} = u_{r1} + u_{r1}.u_{r2} + u_{r2}$	for $u_{r1} < 0$ and $u_{r1} < 0$,
$u_{combined} = \frac{u_{r1} + u_{r2}}{1 - \min \{ \lvert u_{r1} \rvert \text{ and } \lvert u_{r2} \rvert \}}$	otherwise.

Note that the first three are combinations of observational uncertainties only, consistent with logic combinatorics (check!). The fourth one combines observational uncertainty with model uncertainty where the imply is not always the case. The fifth is needed for the case when two observations point at the same conclusion.

As an example of the reasoning process consider the following observations (connectivity is assumed but omitted for clarity):

$fact_1$	$object_1$ is house with $u_1 = 0.8$
$fact_2$	$object_2$ is gasline with $u_2 = 0.7$
$fact_3$	$object_3$ is doubly arrowed line with $u_3 = 0.5$

And the following map maker's rules:

$rule_a$	($object_z$ = house to gasline) implies with u = 0.9 z = doubly arrowed line
$rule_b$	(object_at_one_end = house) .and. (object_at_other_end = gasline) implies x = house to gasline

Then reasoning proceeds as follows:

$reasoning_i$ (object_at_one_end x = house with $u_1 = 0.8$) .and.
(object_at_other_end x = gasline with $u_2 = 0.7$)
implies x = house to gasline with $u_i = 0.7$

$Fact_1$ and $fact_2$ have been inserted in $rule_b$. Then, the .and. combinations leads to the minimum of the two u's to establish the uncertainty in the combined result.

$reasoning_{ii}$ ($object_z$ = house to gasline with $u_i = 0.7$) implies with $u = 0.9$
z = doubly arrowed line with $u_{ii} = 0.63$.

The resulting u_i of reasoning step i has been inserted here. Then, the product of the imply combinatoric gives the resulting certainty of 0.63.

$reasoning_{iii}$ $fact_3$ implies doubly arrowed line with $u_3 = 0.5$
$reasoning_{ii}$ implies doubly arrowed line with $u_{ii} = 0.63$

$Fact_3$ and the result of $reasoning_{ii}$ are both leading to the same conclusion, so the fifth combinatoric should be used here, establishing the certainty of the detected doubly arrowed line as $u_{iii} = 0.815$, higher than each of the individual certainties. So, by combining uncertainties one can gain in certainty indeed.

Mycin as a reasoning engine has several known drawbacks, we will not discuss here. As an alternative the belief networks by Pearl have been successfully proposed, but the types of combinatorics are the same.

Meta reasoning is introduced to indicate where to proceed with the next reasoning step. In many systems, meta-reasoning is hard coded and hence limiting the generality of the system. A proper system would consist of a hierarchy of reasoning layers, the top one be called meta-reasoner. We leave the matter by stipulating its inevitable existence, because if not by meta - reasoning how would a system know where to start and when to stop?

In our example, when reasoning has successfully come to an end, now follows the matching of graphics sign on real world co-ordinates. To that end a transformation matrix is computed transposing each entity given in the co-ordinate system of the map into real world co-ordinate system. This transformation is rarely uniform throughout the map as it should account for cuts and tears, as well as shrinking of the paper and measurement errors. We will not go into this module here after which the process to convert paper maps to electronic files is finished.

4 Some concluding remarks on model - based vision

We have presented two paradigms of including domain knowledge in your vision system: physics models working by the principle of continuous parameter optimisation and AI - models propagating the uncertainty from observations through conclusions on the image. We restrict the concluding remarks to design considerations, parameter tuning, and the thoughts of a combined model.

4.1 Design considerations

In an object oriented system design, reuse and abstraction of the module is the prime concern. Note that through the use of graphical sign detectors we have localised all access to the data fields in the detector modules. They serve as virtual sensors, capable of detecting a specific graphical sign. Such a sign may be fixed in shape and simple, such as a lit, or composed of several layers of detectors where the one layer inputs to the other as was the case with the detector for the house. In addition to localising the access to the data in self-contained detector modules, we have also described a strong separation of reasoning about the map maker's rules from the reasoning about graphical signs. Again, such a separation enables reuse of individual components for similar but different purposes.

In physical modelling, the breaking up of the system in isolated components is far less obvious. In effect, the physical model functions as it yields an integral solutions to the segmentation problem. So, reuse of code is more difficult to achieve in physical model - driven vision. However, the code needed for physical models is generally far less complex. In the AI - model case, software was needed for ink - detection and 20 other detectors of graphical signs as well as reasoning modules. This is typical for a non-trivial rule-based reasoning systems. In a physical model, the number of computations may be very large by the number of iterations required, but there are only a limited aspect to consider in the model and hence write code for.

4.2 Parameter tuning

In the symbolic - model driven approach, we have described the possibility to break up the code in self-contained modules which will have the sole access to the data fields. In addition to be better maintainable, computing modules designed in this way can also be tested and optimised in their parameters in an off-line experimentation environment, searching in a database of well-documented sign examples for the best settings to achieve detection in the particular domain at hand.

In the force model - driven approach, parameter optimisation is intrinsically integral, and in effect part of the computations on each individual image. In fact, parameter optimisation is what the algorithm is after given the data and the abstract model it has at its disposal. Tuning of parameters to improve performance or results can be implemented by confining or directing the search for the best fit in parameter space by using domain specific knowledge.

The take home message is that whereas many have preceded you in applying a symbolic model for a domain where the domain is all continuous and vice versa where a domain is governed by discrete rules to apply a continuous model, you should do so and safe a year in trying to solve an digital vision problem.

5 Literature and further reading

1. P. Shi, G. Robinson, R.T. Constable, A. Sinusas, J.S. Duncan: "A model - based, integrated approach to track myocardial deformation using displacement and velocity constraints", proc. ICCV'95, Cambridge MA, 687 - 693., also accesible through http://noodle.med.yale.edu/.
2. M. Worring, A. W. M. Smeulders, L. H. Staib, J.S. Duncan: " Parameterized feasible boundaries in gradient vector fields.", Computer Vision and Image Understanding, **63 - 1**, 1996, 135 - 144.

3. Special issue on "Model - based vision" in Computer Vision and Image Understanding, volume **61 - 3.**
4. Special issue on "Physics - based modeling and reasoning in computer vision" in Computer Vision and Image Understanding, projected for december 1996.
5. A. Califano, R. Kjeldsen, R.M. Bolle: "Data and model driven multi resolution processing", Computer Vision and Image Understanding, volume **63 - 1**, 27 - 49, 1996.
6. A.W.M. Smeulders, T.K. ten Kate: "Software system design for paper map conversion", In: "Graphics Recognition, methods and applications", (R.Kasturi, K.Thombre, eds.), Springer Verlag, 1995/6, pp. 204 - 211.
7. J. E. den Hartog, T.K. ten Kate, J.J. Gerbrands: "Knowledge - based interpretation of utility maps", Computer Vision and Image Understanding **63 - 1**, 105 - 117, 1996.
8. P.Lucas, L. van der Gaag: *"Principles of Expert Systems"*, Prentice Hall.
9. K. Ng, B. Abramson: *"Uncertainty management in expert systems"*, IEEE Expert 1990.