Using Constraints for the Identification of Buildings in Aerial Images*

Thomas H. Kolbe Lutz Plümer Armin B. Cremers

Institut für Informatik III
Universität Bonn
Römerstr. 164
D-53117 Bonn
{tk,lutz,abc}@cs.uni-bonn.de

Abstract

Aerial images constitute an important data source for Geo-Information Systems. In order to get actual data at reasonable costs, the development of (semi-)automatic tools has been an active research topic in photogrammetry and image processing in the recent years. Based on established techniques for low level syntactic operators such as filters, feature extraction, line detectors and simple pattern matchers, nowadays there is strong interest in explicit models in order to improve the identification of semantically meaningful objects. From the pixel to the object level several representation formalisms such as graphs of extracted image features, aspect graphs and constructive solid geometry (CSG) are applied. Constraint Logic Programming has been identified as an adequate representation formalism and implementation language for building the necessary experimental environment, specifying the models on the different levels, expressing strong heuristics and approaching the complex search problem involved in object detection. This paper focusses on the detection of buildings. It discusses model representation by CLP program fragments and the required adaption and extensions of the CLP(FD) solver of ECLIPSE, the Prolog/CLP platform underlying our implementation. Illustrating examples show how the problems arising in the detection of buildings are approached by CLP techniques.

1 Introduction

This paper describes an application that comes from the research field of object recognition, namely the identification of buildings in aerial images. Three-dimensional building extraction is needed for an increasing number of tasks related to measurement, planning, construction, environment, transportation, energy and property management. Semiautomatic photogrammetric tools are well established [16], but show inefficiencies due to the

*This article appeared in: Proceedings of the 2nd International Conference on the Practical Application of Constraint Technology, London, 1996

extensive amount of data that has to be acquired. So the integration of automatic or at least semi-automatic image understanding tools into photogrammetry seems to be an appropriate way to achieve efficiency in three-dimensional data acquisition.

Due to its high practical relevance, there is much research on this topic in photogrammetry and remote sensing. In fact, there is a research program of the German National Research Council called "Semantic Modeling", bringing together photogrammeters, cartographs and computer scientists. Its main idea is to incorporate "semantics" and "explicit models" into the object recognition process. Another research initiative, funded by the German ministry of research (BMWF), and closely related to our work, aims at constructing a "photogrammetrical eye" in order to improve updates of spatial data for reasonable costs. In this paper we will show how the incorporation of explicit models is done in the subproject "Building Extraction" (of the research program "Semantic Modeling") by using constraint techniques and logic programming. For related work on the relevant CLP techniques see [17, 18, 8, 10], on the systems CHIP [6] and ECLIPSE [11, 12] and on some applications [2].

2 The Role and Use of Models

The structure of the processes of the building reconstruction has been investigated in the early stages of our research and is described in detail in [3]. Figure 1 shows the overall

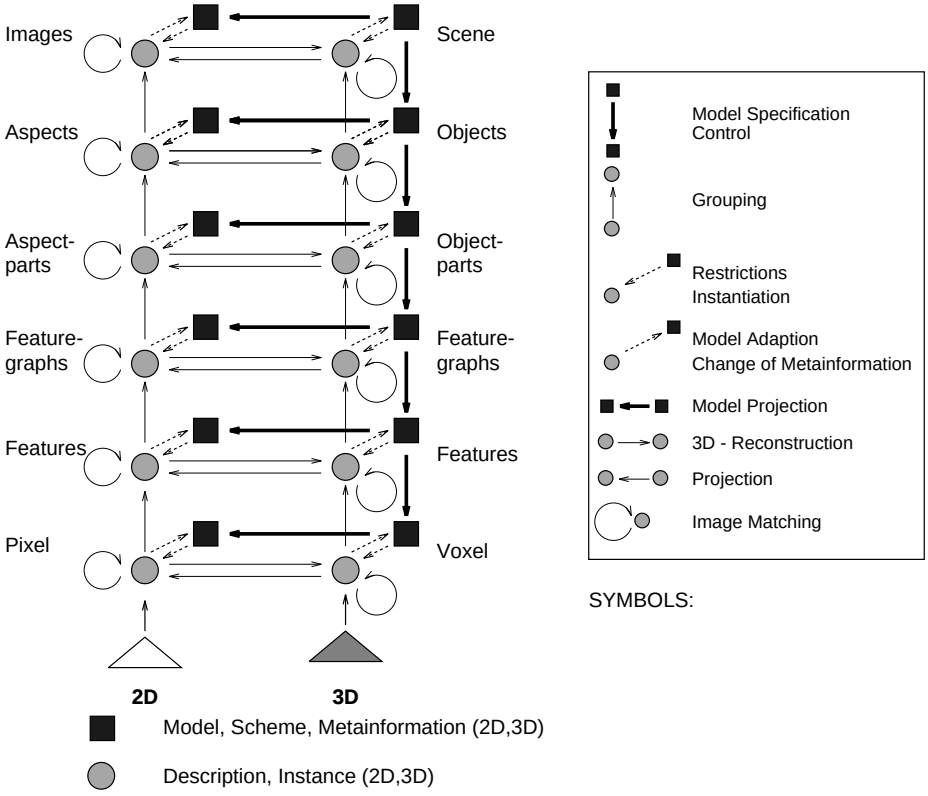


Figure 1: The overall structure for building reconstruction

structure of our concept and illustrates the three main problem dimensions:

- The horizontal axis describes the transition between three-dimensional volumetric objects and two-dimensional image objects. The direction from right to left represents the process of projection defining how objects (and their properties) will appear in the image. The reverse direction stands for the reconstruction of three-dimensional information from two-dimensional observations.

- The vertical axis describes the process of aggregation. For example, on the 2D side it means the aggregation of pixels to features, features to aspects and finally aspects to the complete scene description. In this hierarchy, ascending one or more levels corresponds to a data-driven bottom-up reasoning, while descending means a model-driven top-down reasoning.

- The third axis represents the distinction between model and instance. The direction from back to front means the subsumption of an instance to a class. The other direction stands for the data induced adaption of models in the sense of learning.

The starting point of the interpretation is given by one or more 2D raster images. Additional 3D information about the height of observable objects in the context of the given scene may be provided by a "digital elevation model" [20], if available. The final result will be a three-dimensional scene description (upper right corner). An interpretation strategy now corresponds to a heuristic for finding an appropriate path from the bottom level (left or right side resp. 2D or 3D) to the upper right front node in the graph defined by the "tower model" in figure 1.

The formal methods applied to describe our models are CSG[1]-models augmented by constraints, aspect graphs and structures for image features and their interrelationships.

On the object level volumes, geometry and physical properties of buildings are represented by CSG (constructive solid geometry) models, a formalism well established for man made artefacts in Computer Aided Design (CAD). While CSG models construct complex objects from primitives by a set of boolean operators, most of the semantics of buildings have to be represented by additional constraints. While CSG models allow to say that a simple hip–roof house consists of a block and a prism, additional constraints state, for example, that the prism is fitted to the upper part of the block in a way which takes into account the existence of roof gutters.

On the image level, there are observable features such as points, lines and areas, associated by relations such as parallelism, collinearity and adjacency. These observations, extracted by low level, syntactic operators build the feature relation graph (FRG). The representation of this graph is mainly intensional, as only certain relations, such as the feature adjacency, are defined extensionally.

In order to relate objects, represented by constrained CSG models, to observations, aggregated in FRGs, a third model, aspect graph, has been specified. To each (polyhedral)

[1] Constructive Solid Geometry.

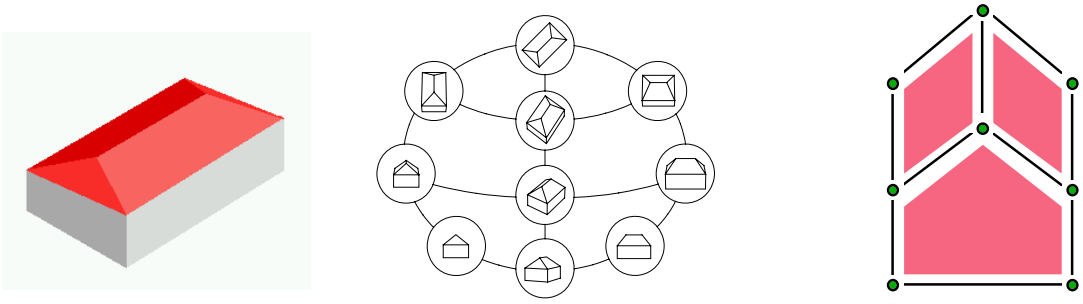


Figure 2: On the left: 3d model of a hip-roof house; in the middle: the corresponding aspect graph; on the right: 2d model of one aspect

CSG model of a house one aspect graph is derived, specifying which part of that model is observable from a certain viewpoint. The nodes of an aspect graph represent the topologically different views (aspects) of 3d objects and the edges specify changes of view (see [5, 4, 14] on the relevance of aspect graphs for computer vision and their computation). Figure 2 shows the CSG model of a building, its aspect graph and an expected image model (under ideal conditions) for one node of that graph consisting of points, lines, and areas.

2.1 Image Model and Data

Aerial images have a typical size of about $10\,000 \times 10\,000$ Pixels. Grey level images have one channel for each pixel, colour or radar images more. In order to allow the interpretation of its contents, the image is segmented and aggregated into three feature classes: points, lines and homogeneous areas. A framework on low level feature extraction is given in [7]. Figure 3 shows a part of an aerial image on the left and the result of its segmentation on

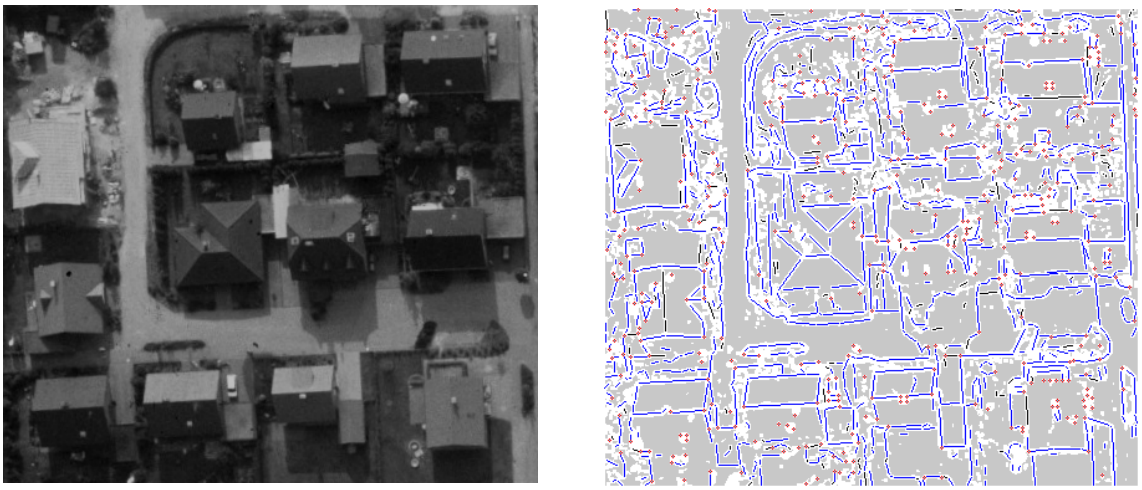


Figure 3: Portion of an aerial image and its segmentation

the right. The extracted features form a level of symbolic description (similar to the one used in [15, 9]) which is transformed to Prolog-terms. Features are stored as Prolog-facts such as

```
line(id,y1,x1,y2,x2,angle,length,scale,type)
point(id,y,x,scale,strength,type)
segment(id,bbox_y1,bbox_x1,bbox_y2,bbox_x2,stripes,size,circum,
        form,holes,ipoint_y,ipoint_x,diameter,gpoint_y,gpoint_x,
        angle,intensity,ivariance)
```

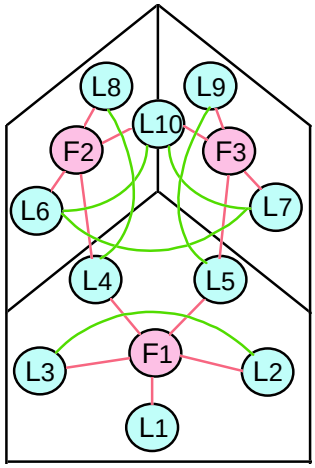
Another result of the segmentation is the feature adjacency relation, enumerating the neighbours of each feature. Being an extensional sub-graph of the FRG, the feature adjacency graph (FAG) is represented by Prolog-facts of the form

```
fag_edge(edge_id,feature_id1,type1,feature_id2,type2)
```

This small portion of an aerial image already has about 3 300 points, 3 100 lines and 2 100 areas. The corresponding feature adjacency graph contains 24 000 edges.

2.2 Object Models and their Transformations

From constrained CSG models we derive the corresponding aspect graphs represented by Prolog-facts similar to the facts representing the image features. Implicit to this transformation is a mapping of three-dimensional object models to two-dimensional representations. Each of these aspects is transformed to a CLP clause as the one in figure 4.




```
house_aspect(F1,F2,F3,L1,...,L10) :-
     blob_domain(DomBlob), [F1,F2,F3]::DomBlob,
     line_domain(DomLine), [L1,...,L10]::DomLine,
     alldistinct([F1,F2,F3]),
     alldistinct([L1,...,L10]),
     fag_arc(F1,L1), fag_arc(F1,L2), fag_arc(F1,L3),
     fag_arc(F1,L4), fag_arc(F1,L5), fag_arc(F2,L4),
     fag_arc(F2,L6), fag_arc(F2,L8), fag_arc(F2,L10),
     fag_arc(F3,L5), fag_arc(F3,L7), fag_arc(F3,L9),
     fag_arc(F3,L10),
     line_parallel(L2,L3), line_parallel(L6,L7),
     line_parallel(L6,L10), line_parallel(L7,L10),
     line_parallel(L4,L8), line_parallel(L5,L9).
```

Figure 4: Representation of an aspect by constraints

As the figure suggests, the number of variables corresponds to the number of features associated with one aspect and the number of constraints to the number of relations specific for that aspect. The domain variables $F_1, \ldots, F_3$ represent expected observable areas,

$L_1, \ldots, L_{10}$ expected observable lines. Their domains consist of the features of the corresponding class extracted from the whole image. Therefore the domain sizes are determined by the number of extracted lines resp. areas. In the listed CLP clause `house_aspect` the domains are constructed and assigned to the variables by the two first lines. The next two lines impose `alldistinct` constraints on each class of variables, enforcing that all variables are assigned different image features. The following `fag_arc` constraints restrict the given variable pairs to have only adjacent features for values. Finally the `line_parallel` constraints state that lines must be parallel in order to constitute valid values for the given variable pairs. The other, mostly geometrical constraints have been omitted here to keep the example simple. Most of the constraints refer to intensional relations of the FRG (i. e. parallelism). That means instead of a simple lookup explicit arithmetic computations have to be done, which degrades performance considerably. Appropriate representation of geometry, however, reduces necessary computations. If, for example, lines are specified by polar coordinates, test for parallelism is immediate. Other representations may require time consuming calculations.

Applications of CLP in domains with spatial attributes, although promising, have been rare. One example, however is the paper of Monfroy [13].

We have found that, among others, the following (binary, symmetrical) constraints are essential for our application context:

- (Approximate) parallelism of lines: `line_parallel`(L_1, L_2) is true, if $|\alpha_{L_1} - \alpha_{L_2}| < \varepsilon$, $\varepsilon > 0$.
- Adjacency of features: `fag_arc`(F_1, F_2) is true, if (F_1, F_2) is an edge of the (extensional) feature adjacency graph (FAG).
- Minimum angle between lines: `line_min_angle`(L_1, L_2, β) is true, if $\neg(L_1 \| L_2)$ and for the angle enclosed by the prolongations L_1' and L_2' of L_1 resp. L_2 holds $\angle(L_1', L_2') \geq \beta$.
- Minimum distance between lines: `line_min_distance`(L_1, L_2, Δ) is true, if the function line_line_dist$(L_1, L_2) \leq \Delta$.

For the simple building types that we are currently investigating, each aspect is typically specified by about 15 variables and 45 constraints.

3 Applying Constraint Logic Programming

Constraint Logic Programming was identified as a language context being able to represent and link the different kinds of models used in the interpretation process. Above, the application context suggests to use the CLP(FD) paradigm for the solution of the difficult combinatorial problems arising in image interpretation. In fact, Waltz' seminal paper on the interpretation of line drawings [19], which initiated the AI-related research on constraint propagation and from which the technique of forward checking underlying CLP(FD) emerged, was entitled "Understanding Line Drawings of Scenes with Shadows".

3.1 Finite Domains of Structured Terms

As described above, the aspect model implies a domain variable for each expected feature in the image. To instantiate this template, thus identifying the corresponding object we are looking for, one has to find a matching set of extracted image features. This is achieved by solving the given CSP, which either fails, if there is no possible solution, or succeeds with a complete and consistent variable binding. The extracted image features are given as facts of a database as the result of the previous image segmentation process. Due to the finiteness of the domain of the objects, we were interested in the application of a CLP(FD) constraint solver. We decided to use the Prolog / CLP system ECLIPSE, because of its good scalability, extensibility, and efficient implementation of domain reduction, propagation, and suspension handling [11, 12].

The three different image feature classes point, line and area are objects of complex structure. They are represented by Prolog-terms (see above). This representation causes a problem, because in most implementations of CLP(FD) variable domains may only contain integer numbers or atoms, thus making it impossible to represent a complex object as one value of a domain variable. To overcome this problem, we define the variable domains D_{blob}, D_{line} and D_{point} as the sets of the (atomic) object *identifiers* of all extracted objects of the corresponding classes. The general feature domain $D_{feature}$ is then defined by $D_{feature} := D_{blob} \cup D_{line} \cup D_{point}$.

3.2 Adapting and Extending the FD-Constraint Solver

Given the problem described above, constraints relate names of the respective terms specifying aggregated objects. Thus the syntactical format is satisfied. The semantics of our constraints illustrated above refer to specific attributes of objects. Parallelism, for instance, refers to the *coordinates* of *points* of *line features*.

Therefore the given standard (and often built-in) constraints of most CLP(FD) systems (i. e. $<$, $\leq$, $=$, $\neq$, $\geq$, $>$, `atmost` and `element`) cannot be applied directly. An extension to the built-in solver has been implemented, which is able to reflect this indirection and to handle the constraints specified in section 2.2. This extension allows to declare arbitrary test predicates (deterministic, all arguments ground when called) to be handled by the constraint solver. Similarly to the lookahead- and forward-declarations of the CLP system CHIP [6, 17], constraints are declared by specifiying the inference rule to be used for solving. So the (approximate) parallelism constraint is implemented in the following way:

```
line_parallel_test(L1,L2) :-
        line(L1,_,_,_,_,Alpha,_,_,_),
        line(L2,_,_,_,_,Beta,_,_,_),
        Diff is abs(Alpha-Beta),
        Diff < 5.
line_parallel(L1,L2) :-
        forward_check(L1:L2:line_parallel_test(L1,L2)).
```

As the listing shows, the indirection is handled by the test predicate, which first fetches the needed object attribute values (here the line angles) from the database and then checks for the condition. The second predicate specifies that `line_parallel(`L_1, L_2`)` has to be handled by the forward checking handler, and that L_1 and L_2 are the domain variables of this constraint. The `line_parallel` constraint is forward-checkable if either both arguments are ground or one argument is ground and the other is a domain variable.

Whereas the number of constraints is rather big, in most cases a single constraint is not very specific. On the other hand, the variable domains are of considerable size. Thus forward checking alone is too weak and requires nearly almost explicit generation of values for variables. The special characteristics of the FAG, mainly the small degree of its nodes, make "look ahead" feasible. It plays a crucial role in our context, if implemented efficiently, as the benchmarks given in the last section illustrate. The efficiency problem of look ahead lies in the fact that while the respective constraints are being evaluated repetitively, computation costs for each single evaluation may be rather high. A special "caching mechanism" allows to replace time consuming computations by simple lookups. The cache is a set of Prolog facts of the form

```
cache(constraint_name, old_domain1, old_domain2,
                       new_domain1, new_domain2, changed1, changed2)
```

where in this example `constraint_name` is a binary predicate and the two flags `changed1`, `changed2` indicate reductions of the respective domains. In a certain sense (`old_domain1`,`old_domain2`) specify a context in which a domain reduction for the given constraint may be iterated.

4 Controlling the Search: Heuristics

The problem of object identification has an immanent combinatorial complexity. Above, the number of involved variables, their domain sizes and the number of constraints is very high, resulting from the magnitude of extracted image data and the large number of features, required to specify buildings, starting from aspects over aspect graphs to all different CSG models. Obviously, the identification of strong heuristics supporting the inherent capabilities of CLP(FD) is of utmost importance.

We apply heuristics in the two (successive) reasoning phases of constraint placement and variable instantiation. In both phases the heuristics are used to determine the order in which the constraints resp. the variables are processed.

Since CLP(FD) has an incremental solver, where the actual constraint store is only updated and not completely rebuilt when new constraints are added, the careful selection of the next constraint to add to the constraint store can help to keep domains small after each step of the placement, thus reducing the overall computation time. For example, if there are three domain variables A, B and C with domains $D_A\{1 \ldots 2000\}$, $D_B\{1 \ldots 2000\}$ and $D_C\{1 \ldots 30\}$ and the two look ahead constraints $\mathsf{cons}(A, B)$ and $\mathsf{cons}(B, C)$, it would

be better to process the second constraint first, because a significant reduction of D_B will be very likely. So if D_B is significantly reduced before processing the first constraint, the computational costs of the first constraint will be a lot less. Although this special heuristic sounds rather obvious, unfortunately it is not supplied in ECLIPSE.

When all constraints are in the constraint store and no further domain reduction (due to constraint propagation) can be done, variables must be explicitly instantiated in order to get a solution. As each instantiation initiates propagation and therefore determines the domain reduction of all directly or transitively related variables, the instantiation order is crucial for the effective complexity either.

Above these heuristics for placement and instantiation, another distinction is meaningful:

- *Application dependent heuristics* consider qualitative information imposed by the application models. For example both the model and our observations show that after feature extraction the lines found are more likely to be fragmented than the areas. So it will be advantageous to start variable instantiation with "area variables" and instantiate "line variables" afterwards.

- *General, application independent heuristics* are based on general measures for the propagation potential of variables and the strength of constraints. ECLIPSE provides already the two instantiation heuristics "most constrained variable" and "first failure". We added the heuristic for constraint placement "smallest domains first", which is of special relevance for look ahead constraints.

5 An illustrating example

Our "Interpretation Laboratory" is an interactive tool for the design and evaluation of models, interpretation strategies and heuristics. This software platform, which is implemented in ECLIPSE-Prolog and TCL/TK, offers immediate access to all system components (such as the extended constraint solver, image and model data, graphical user interface) supporting a step-by-step extension. All kind of data is represented by Prolog-terms and stored as facts in the database and therefore immediately accessible at any time and for all components of the reasoning. The tight coupling to the graphical user interface allows additional to the displaying of results the visualization of the whole reasoning process.

The following table lists some benchmarks done on a Sparc-10 using the environment described above and are rather meant as a hint at the magnitude of processing time. The data come from the left picture in figure 5, emphasizing one match between the model given in figure 4 and the extracted image features. The domain sizes in this example are $|D_{line}| = 1185$ and $|D_{blob}| = 212$. Column t_{plcmnt} shows the time needed for the placement of all 45 constraints (before explicit value generation), column t_{solving} the time for calculating all solutions. All time data are given in seconds.

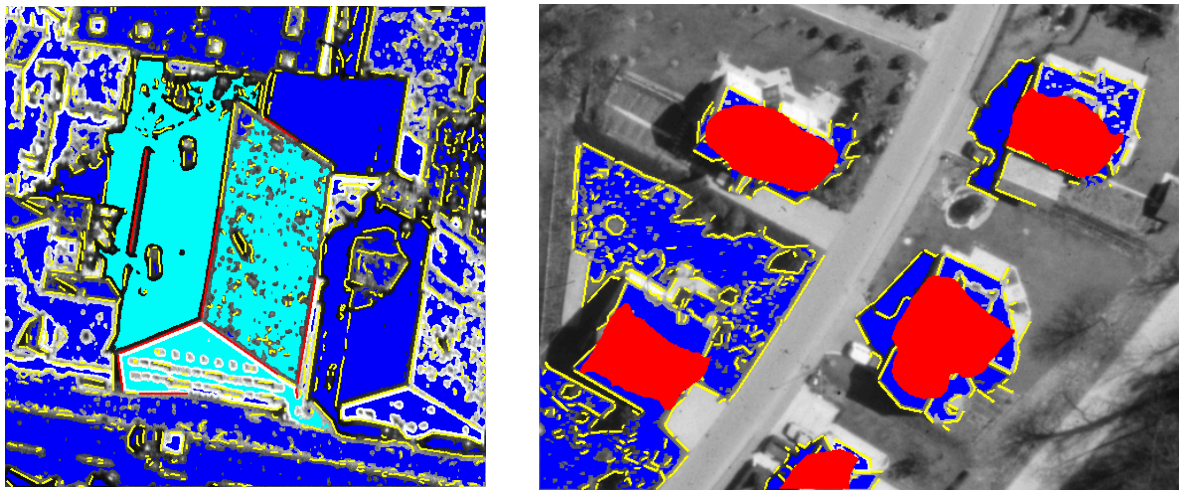


Figure 5: Left: A successful match. Right: Focussing the search by identification of roof areas

Cache	t_{plcmnt}	t_{solving}	Σ_t	# solutions	Cache hit ratio
	without a priori knowledge				
disabled	2.45	22.76	25.21	4608	—
enabled	1.15	17.87	19.02	4608	32.52 %
	with knowledge of roof faces				
disabled	1.11	4.45	5.56	4608	—
enabled	0.77	2.84	3.61	4608	57.55 %
	with knowledge of front face and top edge				
disabled	1.52	1.3	2.82	2160	—
enabled	1.06	0.96	2.02	2160	44 %

The big number of solutions looks rather astonishing at first sight. Apart from symmetries, the fragmentation of one line on the object level into several lines on the image level results in many different possible mappings between the same image part on one hand and the same aspect on the other hand — thus generating distinct "solutions" by our current implementation. Focussing on areas, however, reduces the number of solutions to a figure not much higher than one.

The right picture of figure 5 illustrates how additional information is incorporated in the reasoning process (for the relevance of such information see also [1]) applying the tools of the "Interpretation Laboraty". The spots visualize hypothesis of building positions derived from the analysis of a digital elevation model. These spots focus on the features to be considered in the interpretation process. The benefit of such "a priori knowledge" about possible area features for roof faces reflect in much lesser computation times as shown in the benchmarks.

6 Final Remarks

CLP applications in the past mainly dealt with applications where strict, categorical constraints are appropriate. So far we also focussed on specifying models by such constraints. In reality, however, images are noisy, a phenomenon which has to be dealt with explicitly. Integration of constraint propagation, CLP with techniques representing uncertainty, such as bayesian networks and markov random fields will be a main topic of further research. Above, representation of unobserved features by the incorporation of "wildcards" (more difficult than the corresponding notion of null values in relational databases) and their theoretical underpinning has to be studied.

Acknowledgements

This work was done largely within the project "Semantic Modeling and Extraction of Spatial Objects from Images and Maps" especially in the subproject "Building Extraction" which is supported by the German National Research Council (Deutsche Forschungsgemeinschaft, DFG). We heavily profitted from discussions with our cooperation partners Wolfgang Förstner, Volker Steinhage and Felicitas Lang. We thank the DFG for supporting our work.

References

[1] E. Baltsavias, S. Mason, and D. Stallmann. Use of DTMs/DSMs and orthoimages to support building extraction. In A. Gruen, O. Kuebler, and P. Agouris, editors, *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, pages 199–210. Birkhäuser Verlag, Basel, 1995.

[2] C. Beierle and L. Plümer, editors. *Logic Programming: Formal Methods and Practical Applications*. North-Holland, Amsterdam, 1995.

[3] C. Braun, T. H. Kolbe, F. Lang, W. Schickler, V. Steinhage, A. B. Cremers, W. Förstner, and L. Plümer. Models for photogrammetric building reconstruction. *Computer & Graphics*, 19(1), 1995.

[4] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld. 3-d shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):174–198, February 1992.

[5] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld. From volumes to views: An approach to 3-d object recognition. *CVGIP: Image Understanding*, 55(2):130–154, March 1992.

[6] M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The constraint logic programming language CHIP. In ICOT, editor, *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 693–702, 1988.

[7] W. Förstner. A framework for low level feature extraction. In J.-O. Eklundh, editor, *Computer Vision, ECCV '94, Vol. II*, number 801 in Lecture Notes in Computer Science, pages 383–394. Springer-Verlag, 1994.

[8] T. Frühwirth, A. Herold, V. Küchenhoff, T. le Provost, P. Lim, E. Monfroy, and M. Wallace. Constraint logic programming — an informal introduction. In G. Comyn et al, editor, *Logic Programming in Action*, number 636 in Lecture Notes in Computer Science. Springer, September 1992.

[9] E. Gülch. A knowledge based approach to reconstruct buildings in digital aerial imagery. In L. W. Fritz and J. R. Lucas, editors, *Proceedings of the XVII ISPRS Congress in Washington D. C.*, volume 29, Washington, 1992. ISPRS Commission II, Committee of the XVII Intern. Congress for PRS.

[10] J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 1994.

[11] M. Meier, A. Aggoun, D. Chan, P. Dufresne, R. Enders, D. Henry de Villeneuve, A. Herold, P. Kay, B. Perez, E. van Rossum, and J. Schimpf. SEPIA - an extendible prolog system. In *Proc. of the 11th World Computer Congress IFIP '89, San Francisco*, August 1989.

[12] M. Meier and J. Schimpf. An architecture for prolog extensions. In *Proc. of the 3rd Int. Workshop on Extensions of Logic Programming, Bologna*, 1992.

[13] E. Monfroy. Specification of geometrical constraints. Technical Report 31, ECRC München, 1992.

[14] V. Steinhage. Occlusions and special views within the reconstruction of polyhedral scenes. In *5th Intern. Conf. on Computer Analysis of Images and Patterns, CAIP '93*, number 719 in Lecture Notes in Computer Science, 1993.

[15] J. Stokes. Parsing segmented images. In *Proceedings of the XVII ISPRS Congress in Washington D. C.*, volume 29, 1992.

[16] P. Suetens, P. Fua, and A. J. Hanson. Computational strategies for object recognition. *ACM Computing Surveys*, 24(1):5–61, March 1992.

[17] P. van Hentenryck. A theoretical framework for consistency techniques in logic programming. In *Proceedings of IJCAI'87 Milan, Italy*, 1987.

[18] P. van Hentenryck. *Constraint Satisfaction in Logic Programming.* Logic Programming Series. MIT Press, Cambridge, MA, 1989.

[19] D. L. Waltz. Understanding line drawings of scenes with shadows. In P. H. Winston, editor, *Psychology of Computer Vision*, pages 19–91. McGraw-Hill, New York, 1975.

[20] U. Weidner and W. Förstner. Towards automatic building extraction from high resolution digital elevation models. *ISPRS Journal*, 50(4):38–49, 1995.