Efficient Subgraph Isomorphism with 'A Priori' Knowledge

Application to 3D Reconstruction of Buildings for Cartography

Frank Fuchs and Hervé Le-Men

Institut Géographique National (IGN)
2-4, av. Pasteur, F-94165 Saint-Mandé Cedex, France
Phone : (33)1.43.98.80.66, Fax : (33)1.43.98.85.81
frank.fuchs@ign.fr

Abstract. In this paper, a procedure which computes error-correcting subgraph isomorphisms is proposed in order to be able to take into account some external information. When matching a model graph and a data graph, if the correspondance between vertices of the model graph and some vertices of the data graph are known 'a priori', the procedure is able to integrate this knowledge in an efficient way.
The efficiency of the method is obtained in the first step of the procedure, namely, by the recursive decomposition of the model graph into subgraphs. During this step, these external information are propagated as far as possible thanks to a new procedure which makes the graphs able to share them.
Since the data structure is now able to fully integrate the external information, the matching step itself becomes more efficient.
The theoretical aspects of this methodology are presented, as well as practical experiments on real images. The procedure is tested in the field of 3-D building reconstruction for cartographic issues, where it allows to match model graphs partially, and then perform full matches.

Keywords : Graph matching, error-tolerance, external information, 3-D building reconstruction, cartography, stereoscopy.

1 Introduction

This research work[1] is part of a project aiming at automating the process of 3-D reconstruction of buildings from high-resolution stereo-pairs. The goal is to reconstruct the structure of the roofs. The strategy is model driven. By now, the model corresponding to one building is supposed to be known (some models are shown in Fig. 5).

The models are attributed relational graphs, each vertex standing for a 3-D feature (a 3-D line segment, or a 3-D planar region, or a facade of a building),

[1] This work is supported by the french national cartographic institute, IGN.

F.J. Ferri et al. (Eds.): SSPR&SPR 2000, LNCS 1876, pp. 427–436, 2000.

each edge standing for a geometric property (such as parallelism, orthogonality, ...). The building reconstruction is based on the computation of a subgraph isomorphism between a model G and a graph G_I built on a set of 3-D features derived from the images. Since under detection and geometrical errors are hard to avoid, the subgraph isomorphism computation has to be error-tolerant. Among the existing graph matching techniques ([3][10][13]), the error-correcting subgraph isomorphism detection (ECD) presented in [10] is well suited in our case. More precisely, the shapes of the buildings have usually common subparts, and the ECD is able to take benefit of the common subparts of the model graphs in order to reduce the combinatorial complexity of the problem. This is why the ECD is used in this work to match the model and the data. The ECD uses the concept of edit distance functions, and proposes an efficient method to find the subgraph isomorphism which minimizes an edit distance. This work extends the ECD, so that some major points of the ECD will be first recalled in Sect. 2. However, for more details about the ECD, and the edit distances, see [10], [11].

In this paper, the key point is a modification of the ECD in order to take benefit of an external information (e.g. a user input or a pre-computed information). More precisely, if the correspondance between some vertices of the model and some vertices of the data is already known before the matching, the search space of the matching problem can be pruned by integrating the external information in the main data structure of the ECD, which is called *decomposition*. This integration has to be done carefully, as shown in Sect. 3.

Experimental results about the use of partial matches as external information for the matching procedure will be described in Sect. 4.

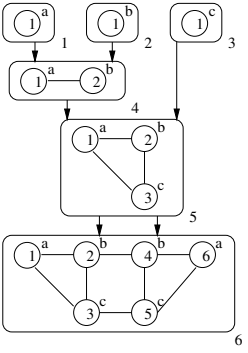
2 Notations

This section mainly recalls some notations used in the ECD ([10], [11]).

- $G = (V, E, \mu, \nu)$ or $G = (V, E)$ denotes a graph. $v \in V$ is a vertex, $e \in E$ is an edge. $\mu : V \mapsto L_V$ is the vertex labelling function, and $\nu : E \mapsto L_E$ is the edge labelling function.
- A subgraph of G based upon a subset of vertices $V_S \subset V$ is denoted $G_{V_S} = (V_S, E_S, \mu_S, \nu_S)$, and the difference of two graphs is $G - G_{V_S} = G_{V \setminus S}$.
- The union of two graphs $G' = G_1 \cup_E G_2$ according to a set of edges E and an edge labelling function $\nu : E \mapsto L_E$ is defined by : $V' = V_1 \cup V_2$; $E' = E_1 \cup E_2 \cup E$; $\mu(v) = \mu_1(v)$ if $v \in V_1$ or $\mu(v) = \mu_2(v)$ if $v \in V_2$; $\nu(e) = \nu_1(e)$ if $e \in E_1$, $\nu_2(e)$ if $e \in E_2$, $\nu(e)$ if $e \in E$.
- A subgraph isomorphism from G to G' is a function $f : V \mapsto V'$ such that f is an isomorphism from G to $G'_{f(V)}$
- A decomposition D is a set of nodes, each node being a 4-*tuple* (G, G', G'', E), also denoted (G, G', G''), such that :
  - $G' \subset G$, $G'' \subset G$, and $G = G' \cup_E G''$: G can be split into its two sons.
  - If $(G, G', G'', E) \in D$, there is no other (H, H', H'', F) such as $G = H$: G appears only once in the decomposition.

- If $G'(resp.G'')$ has more than one vertex, then there is a node (H, H', H'', F) in D such that $H = G'$ (resp. G'').
- If $G'(resp.G'')$ has only one vertex, then there is no node (H, H', H'', F) in D such that $H = G'$ (resp. G'').

For a node (G, G', G''), G will be called the main graph.
Such a decomposition is a network. Each main graph of a decomposition can be recursively split into subgraphs, in a unique way. Figure 1 is an example. On this figure, each rounded rectangle represents the main graph G of a node (G, G', G'') (E is not represented). Each rectangle is pointed by two arrows which come from the two sons G' and G'' of the node. The figure uses 3 labels (a, b, c), and each vertex is identified by a number. The nodes themselves are identified by a number next to their bottom-right corner.
One can remark that the graph 6 is made of two subgraphs (1,2,3) and (4,5,6) which are isomorphic, so that the two sons of this node is node 5, which is stored only once. In [10], Messmer presents a method to insert a model graph into the decomposition. This method is recalled Fig. 1.



Decomposing G into a decomposition D :

1. In D, compute the largest graph S_{max} such that S_{max} is a subgraph of G, and such that there is a node (S_{max}, G', G'', E) in D.
2. If S_{max} and G are isomorphic (G has already been decomposed), then exit.
3. If no S_{max} was found, then choose S_{max} as a random subgraph of G, and decompose S_{max}.
4. Decompose $G - S_{max}$.
5. Add the node $(G, S_{max}, G - S_{max}, E)$ where E is such that $G = S_{max} \cup_E (G - S_{max})$.

Fig. 1. Left : a decomposition. Right : the standard decomposition algorithm.

If the correspondance between some vertices and the data is well established before the matching step, the method is not able to take this knowledge into account. In the next section, a modification of the method in order to use this knowledge is proposed. More precisely, we will add some labels to the model vertices, and modify some definitions of the graph theory, and then propose a new decomposition procedure, which propagates this knowledge throughout the decomposition.

3 Introducing External Information in the Decomposition

Here, the aim is to integrate some *a priori* information about the matching. Namely, we consider that for some nodes of a model graph G, the matching with the nodes of the input graph is already known : for example, the vertex 4 (resp.

5) of graph 6 of Fig. 1 is known to correspond to the vertex 17 (resp. 23) of the input graph. We will first present a first approach to take this into account in the decomposition process. Then, a decomposition method leading to efficient match will be proposed.

3.1 First Approach

The first approach is to add some labels to the vertices of the model graphs in order to use the external information. Formally, the *graphs with a priori matches* (AP-graphs) are defined :

- An AP-graph is a graph $G^{ap} = (V, E, \mu, \nu)$ where $L_V^{ap} = L_V \times (\{\diamond, \$\} \cup V_I)$ where V_I is the set of vertices of the input graph.
  For a vertex $v \in V$, we have $\mu(v) = (l, l_a)$ where :
  - $l_a = \diamond$ means that there is no a priori match for v. This vertex will be called a *strict* vertex.
  - $l_a = \$$ means that the vertex is not matched with any vertex of the input data. This may happen if a partial match of graph G has already shown that v has no correspondance in the data G_I.
  - $l_a = v_i$ (with $v_i \in V_I$) means that v is matched with node v_i of G_I.

  With this definition, an AP-graph can also be written as $G^{ap} = (V^s \cup V^{ap}, E)$, where $V^s \subset L_V \times \{\diamond\}$ and $V^{ap} \subset L_V \times (\{\$\} \cup V_I)$. V^{ap} (resp. V^s) is the set of the vertices of G^{ap} which are (resp. which are not) a priori matched.
- The definition of an AP-graph isomorphism between $G^{ap} = (V = V^s \cup V^{ap}, E)$ and $G'^{ap} = (V' = V'^S \cup V'^{ap}, E')$ is the natural extension of the isomorphism of two graphs. However, the equality between two labels of L_V^{ap} has to be defined : two labels (l, l_a) and (l', l'_a) are equal if and only if $l = l'$ and $l_a = l'_a$. This means that to AP-vertices are isomorphic if their labels are the same and if they share the same external information (i. e. if they are matched with the same input vertex).
  Thus, when there is a isomorphism f between G^{ap} and G'^{ap}, we have $f(V^s) = V'^s$ and $f(V^{ap}) = V'^{ap}$.

In our example, according to this definition, the problem is to match G^{ap} with the data, where the label of vertex 4 is (B, v_{17}), and the label of vertex 5 is (C, v_{23}). This graph is shown in the node 6 of Fig. 2.

According to these definitions, the decomposition procedure proposed in [11] (Fig. 1) still works and takes the a priori matches into account. More precisely, when looking for the largest subgraph of the model (step 1 of the algorithm of Fig. 1), the graph 5 is found. The difference between graph 6 and graph 5 is then graph 7 which has to be split recursively. The largest subgraph of graph 7 is graph 1, so that we now have to split graph 8 into 2 parts. Since 8 is based on 2 vertices which are both a priori matched, the method stops just after having inserted graph 9 and 10 in the decomposition.

One can see that the edges between vertices 2 and 4 (and between vertices 3 and 5) are forgotten very early in the decomposition process. But it is clear that when finding matches for graph 5 (which is the subgraph (1,2,3) of graph 6), it would be useful to take advantage of this edge which holds much information,

namely : the vertex 2 of graph 5 can only be matched with a vertex which is a neighbor of the vertex 17 of the input. This is the key point leading to the pruning of the search space. However, in order to make the matching process of the ECD able to use this information, the information has to be inserted in the whole decomposition. The next section shows that it is possible to use these informations all through the decomposition process, which leads to an efficient matching procedure.

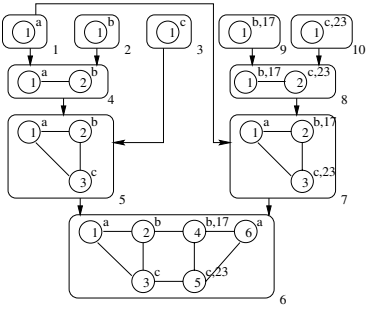


Fig. 2. Example of a decomposition using the AP-subgraph isomorphism

3.2 Efficient Knowledge Usage

The union of two AP-graphs has to be redefined :

- The union of two AP-graphs $G_1^{ap} = (V_1^s \cup V_1^{ap}, E_1)$ and $G_2^{ap} = (V_2^s \cup V_2^{ap}, E_2)$, according to a set of edges $E \subset V_1 \times V_2 \cup V_2 \times V_1$ is the AP-graph $G^{ap} = (V, E)$ such that : $V = (V_1^s \cup V_2^s \cup V_1^{ap} \cup V_2^{ap}, E_1 \cup E_2 \cup E)$. The only difference here is that we do not assume that the intersection between V_1^{ap} and V_2^{ap} is empty. More precisely, if two graphs share some a priori vertices, then these vertices are merged together.
  This definition is important here : since the union of AP-graphs is defined for two graphs which share some vertices, the decomposition process will be able to propagate some shared information between both sons of a node of the decomposition, as shown in the next definition.
- For a graph $G^{ap} = (V^s \cup V^{ap}, E)$ and a subgraph $G_S^{ap} = (V_S, E_S)$ of G^{ap}, we define the extension of G_S^{ap} in G^{ap} as the following : $E(G_S^{ap}) = G^{ap}_{V_S \cup E(V_S)}$ where $E(V_S) = \{v \in V^{ap} | \exists v' \in V^s$ such that v and v' are neighbors $\}$: $E(V^s)$ is the set of AP-vertices of G that are neighbors of at least one strict vertex of the subgraph G_S^{ap}. Thus, $E(G_S^{ap})$ is the graph $E(G_S^{ap})$ itself, plus the vertices that are on its neighborhood, and that hold some external information. As an example, the graph 9 of Fig. 3 is the extension of graph 5 in graph 6, since the two vertices 4 and 5 are neighbors of the nodes 2 and 3 respectively. One can notice that G_S^{ap} is a subgraph of $E(G_S^{ap})$, and that $E(G_S^{ap})$ is a subgraph of G^{ap}.

3.3 The Decomposition Algorithm

The new decomposition algorithm propagates the a priori knowledge through the decomposition. There is only one major change in the graph decomposition. Namely, when the largest subgraph S^{ap}_{max} of a graph G^{ap} to be decomposed is found, we try to see which external information could be added to S_{max}. This is done by extending S^{ap}_{max} in G^{ap}. However, $E(S_{max})$ may not appear in the decomposition. This is why it has to be added first. Moreover, there is a second trick : $E(S^{ap}_{max})$ may be isomorphic to G^{ap} itself : in that case, G^{ap} is made of S^{ap}_{max} plus some a priori vertices. This means that the decomposition of the strict part of G^{ap} has already be done once, but without any 'a priori' information. In that case, we just redo its decomposition, but with external information. This means : choose S_{max} as one of the sons S'_{max} of S_{max}.

Once that the first son of G^{ap} is found, the second one is its complementary $G^{ap} - S_{max}$. Again, we extend this graph in order to complete it with some external information. That way, the a priori knowledge is propagated through the decomposition. It leads to the following algorithm :

1. In D, compute the largest graph S^{ap}_{max} such that S^{ap}_{max} is a subgraph of G^{ap}, and such that there is a node $(S^{ap}_{max}, G', G'', E)$ in D.
2. Let S'^{ap}_{max} be the extension of S^{ap}_{max} in G^{ap}.
3. If S'^{ap}_{max} is isomorphic to G^{ap} then let S'^{ap}_{max} be H' where $(S'^{ap}_{max}, H', H'', E)$ is in D, otherwise, add S'^{ap}_{max} into D.
4. Let $S''^{ap}_{max} = E(G^{ap} - S'^{ap}_{max})$.
5. Add S''^{ap}_{max} into the decomposition.
6. Add $(G^{ap}, S'^{ap}_{max}, S''^{ap}_{max})$.

In the case of graph 6 of Fig. 3, the largest subgraph is graph 5. Now, graph 5 is extend in graph 6. The result of this extension if graph 9, that is put in the decomposition. The best subgraph of graph 9 is again graph 5 (because graph 9 is an extension of graph 5 that we have just computed), so that graph 4 (a son of graph 5) becomes the best candidate. Its extension is graph 10 which then has to be put into D. The whole run of the method leads to Fig. 3, which is more complex, but which fully integrates the external information.

3.4 Analysis of the Algorithm Complexity

This section analyses the complexity of the matching process which takes place after the decomposition itself [2]. The worst case complexity of the error-correcting subgraph isomorphism detection is $O(Lm^n n^3)$ where L is the number of model graphs, n is the number of vertices of the model graphs, and m is the number of vertices of the input graph (i.e. the number of 3-D features extracted on the images). In our case, since we know the match for $|V^{ap}|$ vertices, we only have to match $q = n - |V^{ap}|$ vertices for a graph, so that the worst case complexity of the first approach is $O(Lm^q q^3)$. In the second approach, in the worst case,

[2] The matching process is not described here, because it is a direct extension of the standard error-correcting subgraph isomorphism computation method.

we also may have to compute all the matchings from the models to the data. In particular the neighborhood between the strict vertices and the AP-vertices may not be useful because the edit distance functions consider edge deletion as a possible edition : in the case of an edge deletion of a strict-AP edge there is no gain because the information held by the AP-vertices is lost.

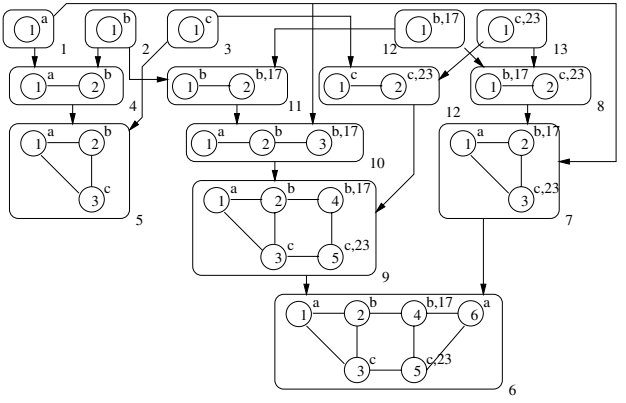


Fig. 3. Example of an efficient decomposition

However, if we do not use the edge deletion as a possible edition (this happens in our application), then let us consider a particular strict vertex v_i. If v_i is not neighbor of an AP-vertex, then it could be matched with any input vertex. But if it is a neighbor of at set of AP-vertices, then the possible candidates for a match with v_i is the set of input vertices which are neighbors of the corresponding input vertices. Let's assume that m_i is the size of this set of vertices, then the worst case complexity of the efficient approach is $O(Lm_1 \dots m_q q^3)$. Depending on the values of m_i, this makes the combinatorics fall drastically down.

4 Application

The results of this study have been applied in the field of building reconstruction, which is of great importance (see [2], [7], [8] for some particular automatic or semi-automatic approaches, and [4], [12] for some collections of articles).

We are working on high-resolution aerial grey-level stereo-pairs (resolution : 8cm / pixel). The building areas are extracted with an automatic procedure which focuses on the raised objects ([1]). These focusing areas are derived from a Digital Elevation Model (a regular grid representing the topographic surface), see Fig. 4 .

After this focusing step, 3-D features are extracted from the images and the DEM. We work with 3 types of features : 3-D lines segments, 3-D planar regions

(see [5]), and 3-D facades. The procedures which compute these features will not be described here. As an example, Fig. 4 (on the right) shows the 3-D line segments.

After the detection step, the input graph G_I is built : an edge (v_1, v_2) is added if v_1 and v_2 have a geometric property (such as parallelism, intersection).

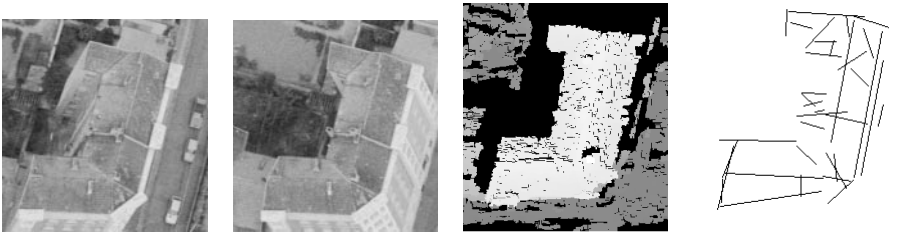


Fig. 4. Initial data : Left Image, Right Image, DEM, and 3-D line segments

The models describe the features we are looking for : their edges hold the geometric properties typical to 3D objects. The matching process is then initiated. It identifies the 3-D features which contribute to the final reconstruction of the object. This procedure has already been described in [6].

Here, we will use 3 different models (see Fig. 5). On this figure, the numbers under each model summarize the size of the model : they correspond to the number of vertices for each type of feature : linear, planar, and facade. The fourth number is the total number of vertices.

The first model (A) is a partial gabbled roof (one side is missing) : this model has 2 surfaces, 5 line segments (1 ridge, 2 gutters, 2 gabled gutters), 3 facades. B is a L-shaped building. C is a L-shaped building with a bevelled corner. It has 26 vertices, which leads to a very combinatoric problem.

The number of vertices for the input graph are : 40 linear vertices, 16 facades, 13 surfaces.

To avoid combinatoric problems for the complex models such as C, we experimented the method and used partial matches. These experiments are summarized in Tab. 1. In this table the rows correspond to the model used, the amount of external information, the CPU time needed with the initial approach, and the CPU needed with the efficient improved approach (the procedure was run on a standard 333 MHz PC).

First, the case 1 shows the CPU times needed for matching model A with the data. It is very low. Second, case 2 shows the matching for a L-shaped (model B) building. The CPU needed is still reasonable. Third, in case 3, the matching for model C leads to a high cost (1 hour), which is not reasonable.

Thus, the experiment 5 is a match of model C with 2 a priori vertices, namely the 2 planar regions found during the match of model A : this external

information is helpful and makes the CPU times fall down. One should notice that the external information in the described case is not a user input but a computed data after a previous match. In case 6, we used the four planar faces extracted from the case 2, which leads to a very low computation time.

After the matching step, the shape of th building is fully reconstructed. The results of the reconstructions of model B and C are presented in Fig. 5 (right). Since model C is a refinement of model B, the use of external information during the match seems useful in a coarse-to-fine approach.

Case	Model	External information	CPU 1	CPU 2
1	partial gabble roof	none	0.5 s	
2	L building	none	5 s	
3	L building with bevelled corner	none	1h	
4	L building	2 planar vertices	5s	1.7s
5	L building with bevelled corner	2 planar vertices	>1h	22 s
6	L building with bevelled corner	4 planar vertices	1h	10 s

Table 1. CPU Times for the matching

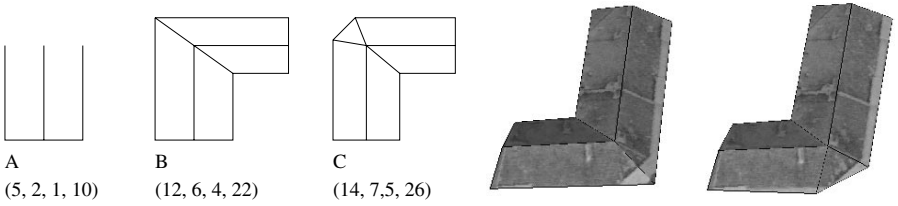


Fig. 5. Left : The 3 tested models. Right : Reconstructions for model B and C.

Future Work and Trends

The objective of this research project is to select automatically the most suitable model to describe a given scene. The simplest way would be, to generate a database of models through the use of rules, to match the models, and then to select the most relevant one. However, once the models are generated, the computation times would be too important, especially for the most complex ones. For this reason, we start by matching *partially* the models first, and then we use the gathered information to extend the models, as shown in the previous section.

Furthermore, the procedure shown here seems to be suitable for a coarse to fine strategy : after having matched a coarse model (such as model B), a better

model can be tested on the input data (such as model C). The main interest here is to avoid testing model C (which is more complicated) if model B is already known to misfit.

5 Conclusion

We have shown that the error-correcting subgraph isomomorphism detection procedures can be extended in an efficient way to integrate some external information. This extension uses an important change in the data structure involved in the method. The method proposed tends to propagate the external information as long as possible during the recursive construction of this structure. Due to this propagation, the resulting matching procedure becomes really more efficient than a naive one, which is ineffective.

This framework is applied in the field of 3-D building reconstruction from aerial stereopairs, where it shows promising results, because it is possible to match partial models and use the results of the matches to match more complex models.

References

1. C. Baillard : Analyse d'images aériennes stéréoscopiques pour la restitution 3-D des milieux urbains. Détection et caractérisation du sursol. PhD Thesis, ENST, Paris.
2. M. Cord : Analyse d'Images Aériennes Haute Résolution : Détection et modélisation du bâti en zone urbaine. PhD Thesis. Université de Cergy Pontoise. 1998.
3. A. D. J. Cross. E. R. Hancock. Graph Matching With a Dual-Step EM Algorithm. IEEE Trans on PAMI. Vol. 20, No 11, November 1998. pp 1236-1253.
4. Computer Vision and Image Understanding. Vol. 72, Number 2, 1998 : Volume dedicated to building reconstruction.
5. F. Fuchs, H. Le Men. Detecting planar patches in urban scenes. Visual Information Processing VIII. Proceedings of SPIE. Orlando. 1999.
6. F. Fuchs, H. Le Men. Building Reconstruction on Aerial Images Through Multi-Primitive Graph Matching. Proceedings of the 2^{nd} IAPR Workshop on Graph-based Representations. Vienna. 1999.
7. A. Gruen, Xinhua Wang. CC-Modeler : a topology generator for 3-D city models. ISPRS Journal of Photogrammetry & Remote Sensing. Vol. 53, 1998, pp 286-295.
8. N. Haala, C. Brenner. Virtual City Models from Laser Altimeter and 2D Map Data. Photogrammetric Engineering & Remote Sensing. Vol. 65, No.7, july 99, pp 787-795.
9. J. -W. Lee and I.-S. Kweon. Extraction of line features in a noisy image. Pattern Recognition, Vol. 30, No 10, October 1997
10. B. T. Messmer. Efficient Graph Matching Algorithms for Preprocessod Model Graphs. PhD Thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, Switzerland, 1996.
11. B. T. Messmer., H. Bunke. A New Algorithm for Error-Tolerant Subgraph Isomorphism IEEE Trans. on PAMI, vol. 20, No 5, May 1998.
12. Proceedings of the Photogrammetric Week 99 : Institute for Photogrammetry Stuttgart University. Stuttgart, Sep 1999.
13. R. C. Wilson, A. N. Evans, E. R. Hancock. Relational Matching by Discrete Relaxation. Proceedings of the British Machine Vision Conference '94.