A Map Mosaicking Method Using Opportunistic Search Approach with a Blackboard Structure

Jonghyon Yi, Min Suk Lee, and Jaihie Kim

Department of Electrical and Computer Engineering, Yonsei University
Shinchon-dong, Seodaemoon-gu, Seoul, 120-749, Korea
{johnyi,lms}@seraph.yonsei.ac.kr
jhkim@bubble.yonsei.ac.kr

Abstract. Map mosaicking is to integrate two or more map images having a coincident area by computing the rotational angle, the vertical and horizontal distances a map image has to move to overlap the coincident area. A solution of the problem is represented as a point in the parameter space with three axes: one for the rotational angle and the others for the vertical and horizontal distances. We extract local features from each map image, match them to make feature pairs, and project the feature pairs onto the parameter space. Traditional approaches using parameter spaces have suffered from a huge search space and computing time, for they project all the feature pairs onto the parameter space and search solutions by iterative optimization methods. We propose a new method that can give a solution not projecting all the feature pairs onto the parameter space but search opportunistically in a Blackboard structure.

1 Introduction

When scanning a large map with a flat bed scanner that can scan only documents of restricted size, one should divide the map into several parts, scan them, and then integrate them by adequate methods. Map mosaicking can be considered as finding coincident areas among these partitioned map images and computing the exact transformations among them. To do this manually, one should find some distinguishing marks on one map image, look for the same marks on another, and then overlap these two coincident marks. To mosaic map images automatically, one can follow the same strategy. In order to reduce computation, we used local features[1], rather than the pixels itself. The features should be consistently obtainable from map images regardless of rotation, scaling, and translation of map images. A feature from one map image and another feature from the other map image form a matched feature pair if they are coincident with each other. A matched feature pair implies a possible transformation between two map images.

Finding the optimal transformation is performed on the set of matched feature pairs. The matched feature pairs are positioned in the parameter space, which has axes of two translational parameters, one rotational parameter and one scaling parameter. The optimal transformation is expected to be at the location where feature pairs densely exist. While early studies tried to find optimal transformation by procedural or mathematical methods[1,2], recent researchers[3,4] addressed that Hopfield neural networks showed good performances in finding the position representing the optimal transformation. All these approaches try to match all possible feature pairs and suffer from heavy computation. Kim *et al*[4]. used Discrete Hopfield neural network and Continuous Hopfield neural network to solve the search space problem.

We propose a new map mosaicking method that can find the transformation solution with a reduced search space by an opportunistic search approach. We first extract local features from each map images, match them, and calculate the similarity between them. Then we compute transformations of matched feature pairs in order of their similarities, and position them on the parameter space as "points". Two matched feature pairs with high similarities closely located in the parameter space imply high possibility of the transformations of the neighboring positions being a solution transformation. These two matched feature pairs trigger the generation of a "cluster" at the middle of these two pairs in the parameter space. The cluster is defined as a candidate for solution transformation and has a region of predefined size. If a new transformation computed from a matched feature pair is positioned close to the center position of a existing cluster, the transformation is included into the cluster. A cluster which has included sufficient transformations is verified to be the solution transformation. These procedures are executed opportunistically through a blackboard structure[5].

We examine the proposed method by applying it to mosaicking Korean Land Register Map scanned with a resolution of 200 dpi. The result shows that the proposed method can mosaic map images in a opportunistic way, and is applicable to mosaicking map images with small coincident areas. In this paper, we assume that there are only rotational and translational transformations between two map images. However, the proposed method can be extended to scaling transformations without fundamental modifications.

2 Map Mosaicking Method Using Clusters in Parameter Space

2.1 Feature Extraction

Features extracted from map images are roads, crossroads, and districts[6]. Figure 1 shows an example map image and the extracted feature information.

A district feature represented by a polygon has several attributes: the lengths of roadsides, the directions of roadsides, the number of neighboring roads, and the widths of every neighboring roads. A road feature represented by a straight line has following attributes: the width of the road, the length of the road, the number of crossroads connected, and the number of connected roads of each crossroad. A crossroad feature has the number of connected roads, the widths of the each connected road, and the angles between connected roads.

A feature has a vector representation for transformation computation. A road feature itself is a line and has a vector represented by two end points. A crossroad

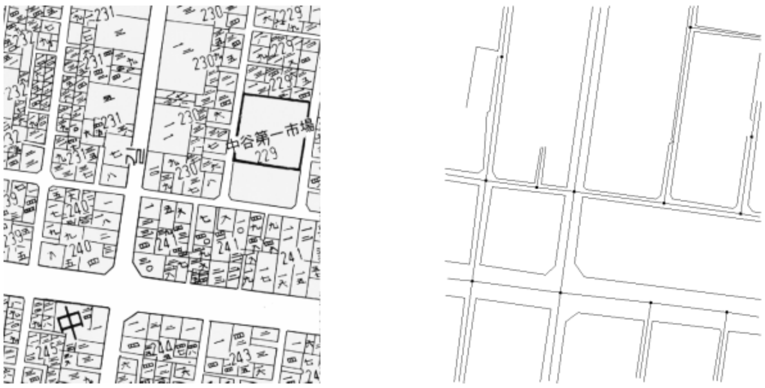
(a) (b)

Fig. 1. An example map image and the extracted feature information.

feature has a vector that starts at the crossroad point and ends at the other end point of the widest connected road line. A district feature consists of several lines. So, any of these lines is used as a vector representation.

2.2 Matching the Features

Two features selected from each map image are matched and the similarity is computed when they are of same type and satisfy one of the three matching criteria. The *coincident* criterion is satisfied when two features fit exactly each other in all attributes. The *partial* criterion means that one feature is a part of the other. Two features sharing part of attributes satisfy the *sharing* criterion.

Crossroad features can be matched only by coincident matching criterion. District features and road features can have all three matching criteria. Figure 2 shows examples of matching criteria: in (a) two point features coincides in all attributes, in (b) one district feature is totally matched to part of the other district feature, and in (c) and (d) two road features share only half of attributes and satisfy sharing matching criterion.

The similarity between features is computed differently depending on the kind of features and the matching criteria but the idea is that the value is between zero and unity, and a pair of similar features has a value close to unity. The similarity can be calculated easily by Eq. 1. In this equation, A_i means the ith attribute of the feature A of the first map image, and B_j means the jth attribute of the feature B of the second map image. *Val(•)* implies the value of an attribute.

$$Sim(A,B) = \prod_{\text{matched attribute } i,\ j} \frac{\min(Val(A_i), Val(B_j))}{\max(Val(A_i), Val(B_j))} . \qquad (1)$$

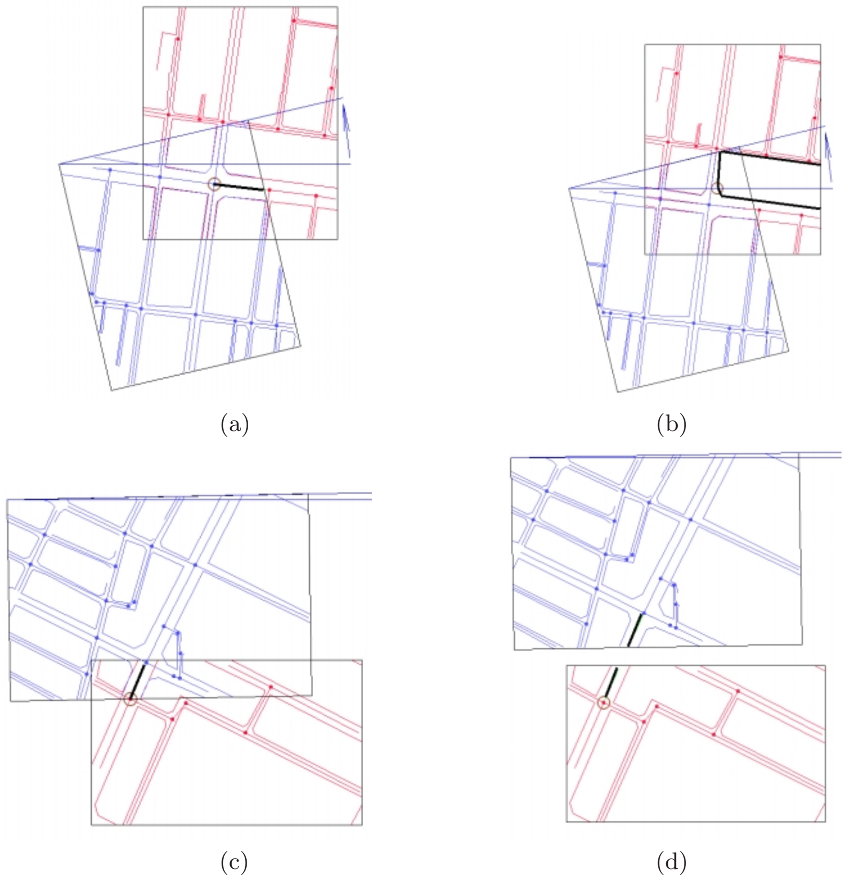

Fig. 2. Examples of matching criteria: (a) coincident criterion of crossroads, (b) partial criterion of districts, and (c) and (d) sharing criterion of roads.

2.3 Computing Transformations from Matched Vector Pairs

Assuming there is only rotational and translational (RT) transformation between two map images, we need a parameter space with three axes: $d\Theta$ axis for rotation angle, dX and dY for horizontal and vertical translation. Then a point ($d\theta$,dx,dy) implies an RT transformation.

For a feature pair, we can compute a transformation between the two features[1]. If two matched features are represented in vector form as $\boldsymbol{v}(x_1, y_1)$, and $\boldsymbol{w}(x_2, y_2)$, the transformation ($d\theta$, dx, dy) is decided as Eq. 2, 3 and 4.

$$d\theta = \angle(\boldsymbol{v}) - \angle(\boldsymbol{w}) \ . \tag{2}$$

$$dx = x_2 - (x_1 \cos d\theta - y_1 \sin d\theta) \ . \tag{3}$$

$$dy = y_2 - (x_1 \sin d\theta + y_1 \cos d\theta) \ . \tag{4}$$

For a feature pair matched by the sharing matching criterion, many possible transformations exist because the two corresponding features can be matched by sliding manner. Figure 2 (c) and (d) show two extreme transformations of this case.

The transformations of feature pairs matched by the coincident and partial matching criteria are computed in order of matching similarity, and represented as points in the parameter space on the corresponding positions. The transformations of a feature pair matched by the sharing matching criterion are represented in the parameter space as a line. The line has two end points at positions corresponding to two extreme transformations as in Figure 2 (c) and (d). A line on the parameter space is a set of possible transformations and it has less information compared with a point representing an exact transformation. So, we compute intersections of the lines to find a transformation with more possibility, and represent them as points in the parameter space.

2.4 Generating Clusters in Parameter Space

Two close points in parameter space, each representing matched feature pair, trigger the generation of a cluster at the middle of these two points. The generated cluster implies a candidate for the transformation solution supported by the two matched feature pairs, and includes these points as a supporting group. It is supposed that majority of points with high similarities are positioned densely at the position of the optimal transformation of the two map images. A cluster generated from two matched feature vector pairs with high similarities has relatively high possibility to be the transformation solution.

It is not preferable to generate too many clusters that are congested closely for the sake of computation efficiency. To prevent this situation, a cluster has a prohibited region of predefined size, and the generation of new clusters is prohibited inside the regions of existing clusters. The size of the prohibited region of clusters determines the precision of the transformation solution search. Figure 3(a) shows an example of generating a cluster from two points in the parameter space. In this figure, the inner circle shaded darkly implies the prohibited region and the outer circle shows the region in which the cluster takes points as its own supporting group.

The cluster c_k to be generated from the two points p_i and p_j has a certainty factor $CF(c_k)$ as Eq. 5. In this equation, $\delta\theta$ and $\delta\tau$ are distances between two points in rotation axis and in translation surface in the parameter space respectively. And $\phi\theta$ and $\phi\tau$ are bias values in rotation axis and in translation surface in the parameter space respectively. The computed certainty factor shows a value closer to unity when two seed points are close to each other and have larger similarities. The certainty factor value determines when the cluster should be generated. The usage of the certainty factor is explained in Chap. 3 in detail.

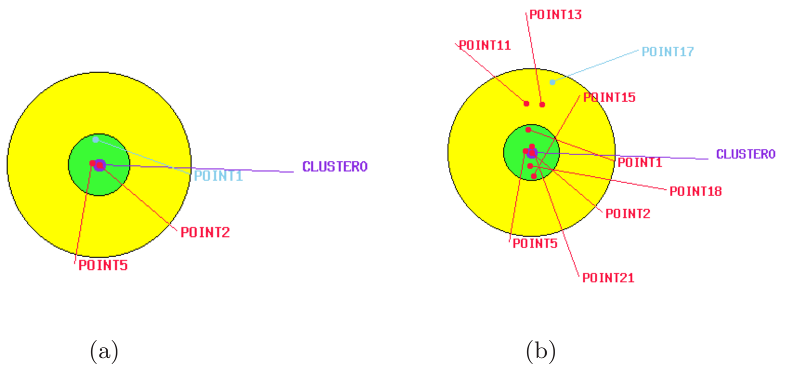

Fig. 3. Examples of a cluster in the parameter space : (a) the generation of cluster CLUSTER0 from point POINT2 and POINT5, (b) the growth of CLUSTER0

$$CF(c_k) = \max\left(Sim(p_i), Sim(p_j)\right) \times \left(1 - \frac{\delta\theta(p_i, p_j)}{\phi\theta}\right) \times \left(1 - \frac{\delta\tau(p_i, p_j)}{\phi\tau}\right) . \quad (5)$$

2.5 Growth of Clusters

If a cluster is generated near a existing point or a new point is positioned near the center of a existing cluster, the cluster can include the point into its supporting group. The probability of a cluster being a solution increases by including points into supporting group. The support value defined below determines whether a point in the parameter space should be included into a cluster or not. The support which point p_i has for cluster c_k is given by Eq. 6. It measures how strong a point supports a cluster. It shows a value close to unity when the certainties of p_i and c_k are large and the distance between them is small.

$$Sup_{p_i}(c_k) = Sim(p_i) \times CF(c_k) \times \left(1 - \frac{\delta\theta(p_i, c_k)}{\phi\theta}\right) \times \left(1 - \frac{\delta\tau(p_i, c_k)}{\phi\tau}\right) . \quad (6)$$

Clusters centered at a position where a large amount of points with high similarities are densely located will grow fast, because points with large similarities and close to centers of the clusters have large supporting values for the clusters and will be included early in the computation.

2.6 Cluster Verification

A sufficiently grown cluster is determined to be a solution transformation through the feature comparison. We call this procedure cluster verification. To decide

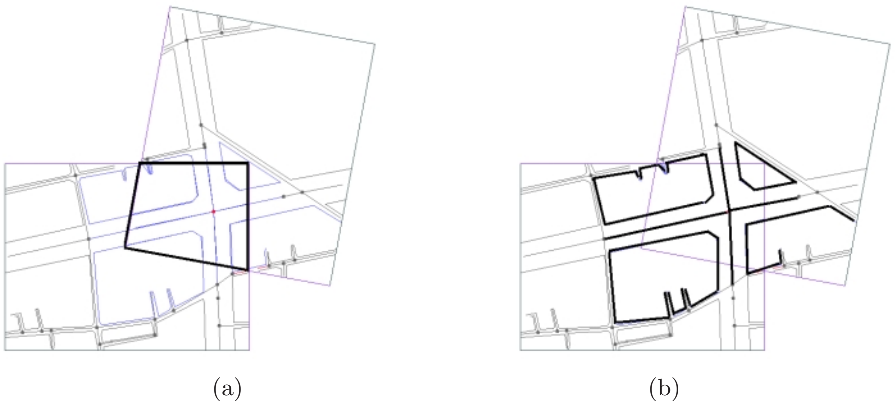

Fig. 4. (a) Composed feature map image and (b) features engaging into the verification

which cluster to verify, we defined the competency. The competency of a cluster c_k is close to unity when the certainty factor of the cluster is close to unity and the supporting group is large as in Eq. 7. Here, $S(c_k)$ is the number of points in the supporting group of c_k and T is the number of matched feature pairs.

$$Comp(c_k) = CF(c_k) \times \frac{S(c_k)}{T} \ . \tag{7}$$

In the cluster verification procedure, competencies of all clusters are compared and the cluster with the largest competency value is selected. To verify a cluster, the features of one map image as Figure 1(b) is rotated and translated onto the features of the other by the transformation designated by the center of the cluster. Overlapped region exists on the composed feature map image as Figure 4 (a). Then, each feature of one map lying on the overlapped region is matched one to one with the corresponding feature of the other map. Verification result is computed with feature pairs matched for verification. Different from the similarity computation in Section 2.2, verification result is computed by Euclidean distance between matched line segments of each matched feature pair, and given as Eq. 8. In this equation, ϕd is a sufficiently large bias value and $d(f_i, g_j)$ is minimal Euclidean distance between feature f_i of one map and g_j of the other.

$$Verif(c_k) = \prod_{\text{matched feature pair } f_i,\ g_j} \left(1 - \frac{d(f_i, g_j)}{\phi d}\right) \ . \tag{8}$$

The search for transformation solution is finished when the cluster verification result is greater than a predefined threshold. For further reliability, one map image is transformed onto the other and pixel level similarity is computed.

3 Opportunistic Search Using Blackboard Structure

The proposed map mosaicking method is implemented by using blackboard structure to take advantage of the opportunistic reasoning characteristics. The hierarchical data structure in the proposed map mosaicking method is composed of map images, features extracted from the map images, matched feature pairs which is represented as points in the parameter space, clusters in parameter space, and the solution transformation. Figure 5 shows the blackboard structure. In this figure, a horizontal line designates a layer of hierarchical data structure and the arrows represent knowledge sources. Each knowledge source takes information from the layer where the arrow starts and puts the results on the layer where the arrow ends.

We use following 5 knowledge sources: *Point Generator*, *Cluster Generator*, *Cluster Grower*, *Cluster Verifier*, and *Stopper*. Each knowledge source is composed of a precondition that determines whether the knowledge source is applicable at the situation, and a body that executes appropriate computation. In our implementation, all preconditions of knowledge sources show their priority values on the basis of current blackboard condition, and the controller of the blackboard system chooses a knowledge source that shows largest priority value. The chosen knowledge source does its job and returns the result to the blackboard.

In the blackboard structure, it is very important to define the priorities of knowledge sources. The point generator takes the greatest value among similarity values of matched feature pairs as its priority value, and the cluster generator takes the certainty factor of the cluster to be generated as its priority value. And the cluster grower takes the largest value among support values that points have for clusters. The priority value of cluster verifier takes into account the number of points in the supporting group of the cluster to be verified and is defined as Eq. 9. In this equation, $CF_{best\ cluster}$ and $S_{best\ cluster}$ are the certainty factor and the number of points in supporting group of a cluster with maximal competency value respectively. The precondition value is close to unity when the certainty factor of the cluster is close to unity and the number of points in supporting group is large. The priority value of the stopper has unity when the cluster verification result is greater than a predefined threshold or when there is no solution even after predefined trial of cluster verification and it has zero otherwise.

$$Precondition_{CV} = (CF_{best\ cluster})^{\frac{1}{K}} \ , \tag{9}$$

$$K = \begin{cases} \frac{S_{best\ cluster}}{4} & \text{when } S_{best\ cluster} \geq 4 \ ; \\ 1 & \text{when } S_{best\ cluster} < 4 \ . \end{cases}$$

When the search for the solution transformation is finished and there is a solution transform, we integrate the two map images by rotating and translating one map image onto the other. The integrated map image may be used for mosaicking with another map image via the same method.

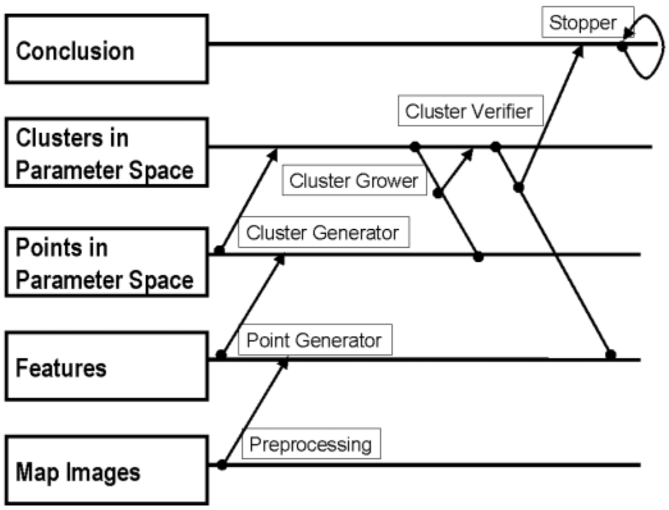

Fig. 5. The blackboard structure for the opportunistic map mosaicking

4 Experimental Results

Korean Land Register Maps are used in the experiment. The map image is scanned with the resolution of 200 dpi and in 256 gray levels. The district information, which is represented in light gray color because it is originally in yellow color, is recognized first. Parallel lines between neighboring districts are recognized as roads. And finally the intersection points of elongated road lines are recognized as crossroads[6,7,8]. The experiment was done on Intel Pentium Pro PC of 200 MHz with the operating system Linux 2.0.

Figure 6, 7 and 8 show the examples of the experiment. In each figures, (a), (c) are map images, and (b), (d) are feature images extracted from (a) and (c) respectively. Using the proposed mosaicking method, the integrated feature image (e) is obtained by the solution transformation, and the mosaicked map image is shown in (f). Characteristics of problems are given in Table 1

The summary of the results of Problem 1, 2 and 3 is shown in Table 2. In Problem 1, one can guess the solution easily because of the outstanding local features such as a crossroad. With the proposed method, it takes a lot of computation time in preprocessing and similarity computation because there are many district, road, and crossroad features despite its small image sizes. And it takes relatively small portion of time in the opportunistic search processing because there are large coincident area and maybe because there are many feature pairs matched with high similarities.

In Problem 2, human may feel it hard to find the solution because of little coincident area. Only 18.0 seconds were taken in preprocessing and similarity computation because of small number of features. Large part of processing time

Table 1. Characteristics of Problem 1, Problem 2 and Problem 3.

Problem1	Image Size	Number of District	Number of Road	Number of Crossroad
Map1	600 x 716	17	26	12
Map2	600 x 716	19	30	13
Problem2				
Map1	808 x 409	9	11	5
Map2	900 x 600	24	35	7
Problem3				
Map1	582 x 732	20	33	18
Map2	782 x 600	23	36	15

Table 2. Experimental results of Problem 1, Problem 2 and Problem 3.

	Problem 1	Problem 2	Problem 3
Solution Transformation ($d\theta$: degree, dx and dy : pixel)	$d\theta$: -13.25 dx: -260.45 dy: 485.30	$d\theta$: 2.21 dx: -172.04 dy: -597.73	$d\theta$: 0.17 dx: 143.18 dy: -239.22
# of Points in Parameter Space	22	53	10
# of Clusters in Parameter Space	4	18	7
# of Clusters Verified	1	7	3
Total Time (TA+TB) (second)	42.2	39.0	44.0
Preprocessing Time + Similarity Calculation(TA) (second)	31.0	18.0	37.0
Time Elapsed in Opportunistic Search(TB) (second)	11.2	21.0	7.0
# of Execution of Knowledge Sources	55	178	28

was spent in opportunistic search phase trying to verify 7 clusters in parameter space. However, one can expect that it will not be possible to find the solution without the sharing matching criterion defined in this paper.

In Problem 3, because there are very large coincident area and many coincident features, it took less time in opportunistic search compared to other examples. One can see this from the fact that there are 10 points and 7 clusters in parameter space. This means that most of the transformations are congested closely.

5 Conclusion

We proposed a map mosaicking method using clusters in the parameter space and the opportunistic search approach. We showed that the proposed method can find the transformation solution with a reduced search space compared to that of conventional methods using parameter spaces. We adopted a blackboard structure in building the system and realizing the opportunistic search method.

By defining "clusters", we were able to manage candidates for transformation solutions, and get a solution transformation with small amount of computation. By using the sharing matching criterion, mosaicking map images that have relatively small coincident area was made possible.

In this paper, we dealt with only two partitioned map images. To integrate more than two images, we can follow the same method discussed previously. First, we select another partitioned map which have coincident area with the previously mosaicked map. Second, we extract features from the selected map. And we finally apply the same method to the newly generated feature map and the formerly composed feature map as in figure 6(e), 7(e) and 8(e). These steps would be continued until all the partitioned map images are integrated together.

Careful readers may find out that our method is not for searching the "optimal" transformation because we define the prohibited region of a cluster of fixed size. However, we can insist that finding suboptimal transformation solution in short time is more helpful for map mosaicking applications than finding optimal solution with large computation because map images are generally large in size. Furthermore the precision required by an application can be achieved by changing the size of the prohibited region of the cluster.

The opportunistic approach used in this paper is applicable to object recognition, image mosaicking, and also to image registration, in case that robust features are available.

Acknowledgments

This work was supported in part by Korea Science and Engineering Foundation by the Contract 95-0100-11-01-3.

References

1. G. Stockman, S. Kopstein, and S. Benett, "Matching Images to Models for Registration and Object Detection via Clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 3, pp. 229-241, May 1982. 322, 323, 325
2. I. Dowman, V. Vohra, and A. Holmes, "Developments in automated object- image registration," *Integrating photogrammetric techniques with scene analysis and machine vision II*, Orlando, FL, pp. 85-92, Apr. 19-21, 1995. 323
3. N. M. Nasrabadi, W. Li, "Object Recognition by a Hopfield Neural Network," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 21, No. 6, pp. 1523-1535, November/December 1991. 323
4. J. H. Kim, S. H. Yoon, and K. H. Sohn, "A Robust Boundary-Based Object Recognition In Occlusion Environment By Hybrid Hopfield Neural Networks," *Pattern Recognition* Vol. 29, No. 12, pp. 2047-2060, 1996. 323
5. Robert Engelmore, Tony Morgan, and H. P. Nii, *Blackboard Systems*, Addison-Wesley Publishing Company, 1988. 323

6. K. S. Jang, J. Yi, J. Y. Jung, J. Kim, and K. H. Chang, “A Recognition of Map Using the Geometric Relations between Lines and the Structural Information of Objects,” *Proc. of IEEE Int’l Conf. on Image Processing*, Santa Barbara, California, pp. 150-153, Oct. 1997. 323, 330
7. Mark J. Carlotto, “Using maps to automate the classification of remotely- sensed imagery,” *Algorithms for multispectral and hyperspectral imagery II*, Orlando, FL, Apr. 9-11, 1996 (A96-42073 11-35), SPIE Proceedings. Vol. 2758, pp. 40-50, 1996. 330
8. L. Boatto, V. Consorti, M. D. Buono, S. D. Zenzo, V. Eramo, A. Esposito, F. Melcarne, M. Meucci, A. Morelli, M. Mosciatti, S. Scarci, and M. Tucci, “An Interpretation System for Land Register Maps,” *IEEE COMPUTER*, pp. 25-32, July 1992. 330

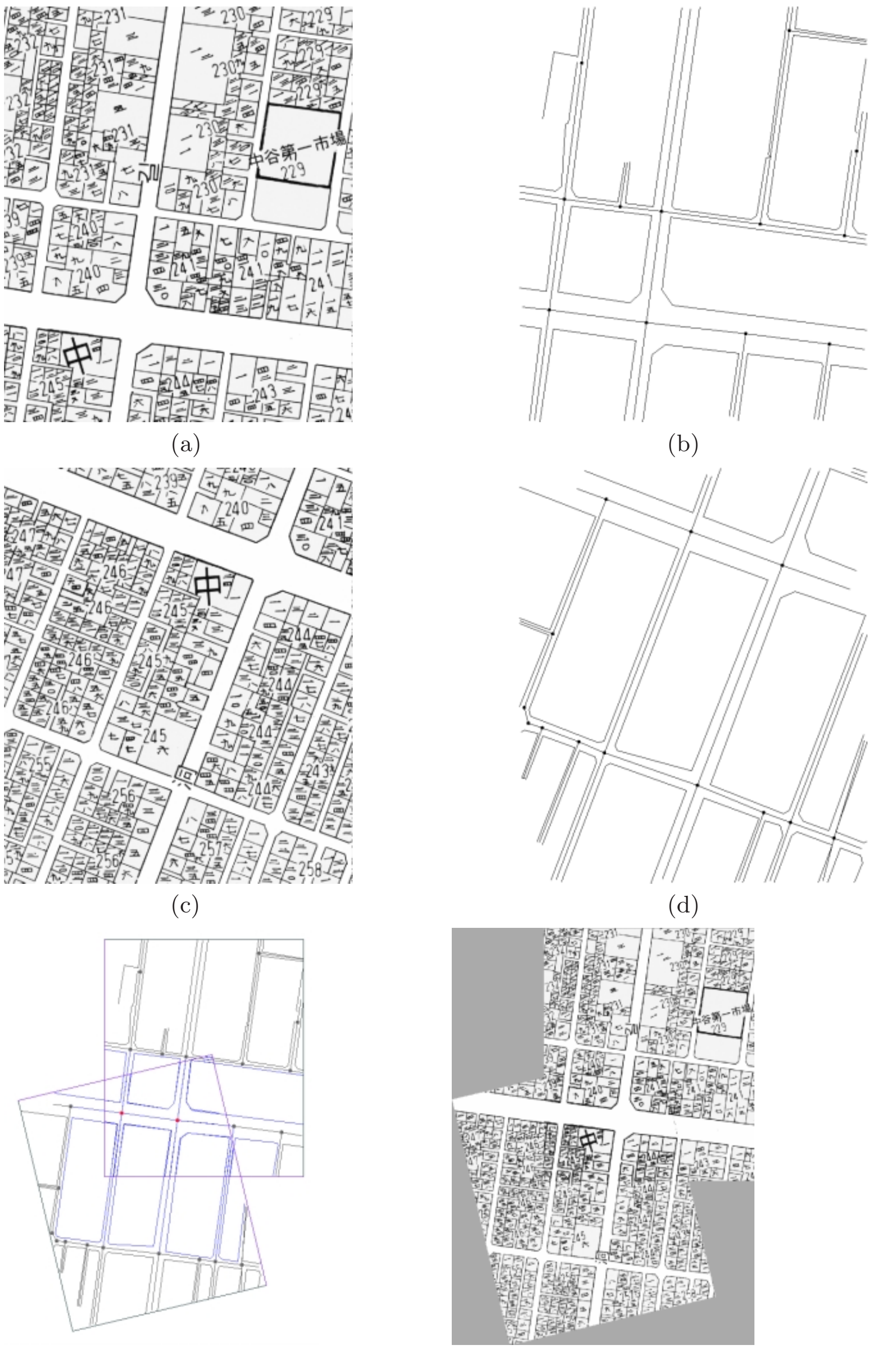

Fig. 6. Examples of the experiment : Problem 1

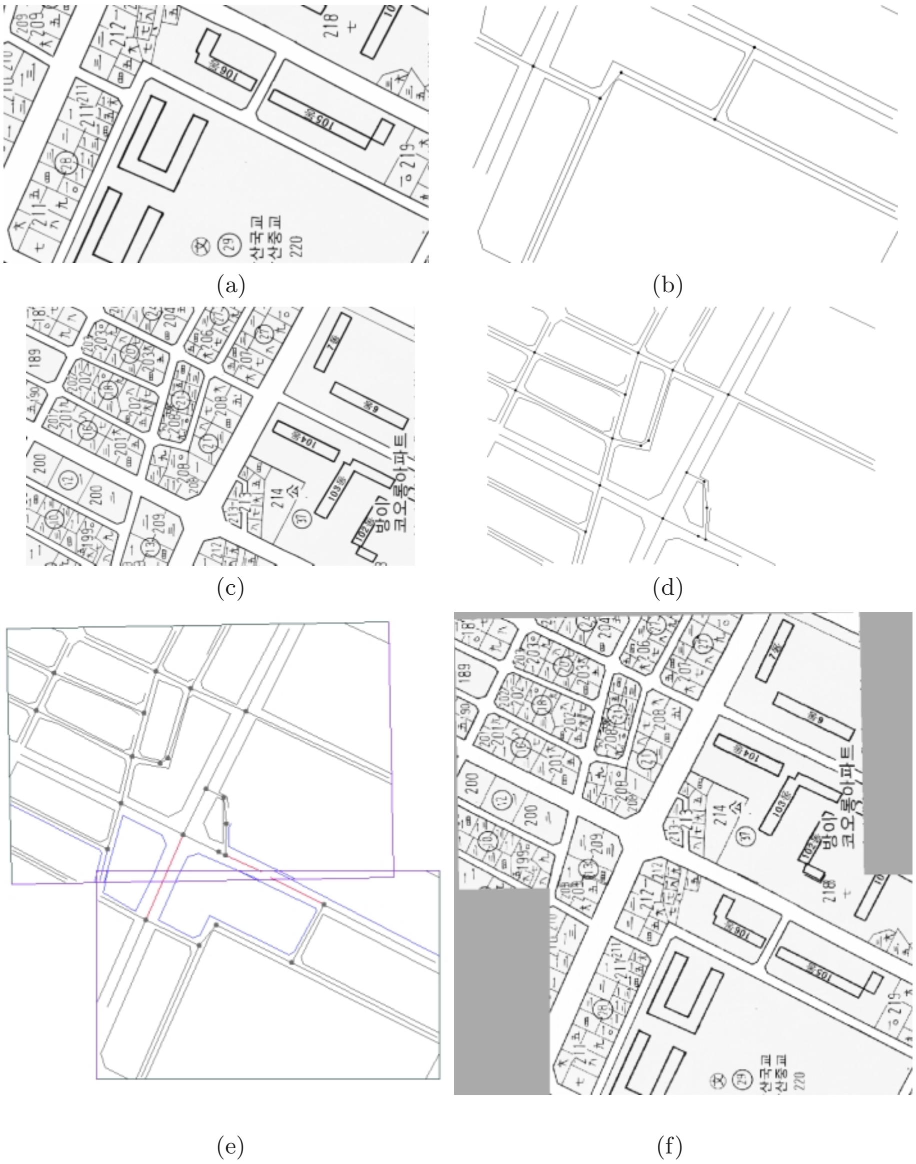

Fig. 7. Examples of the experiment : Problem 2

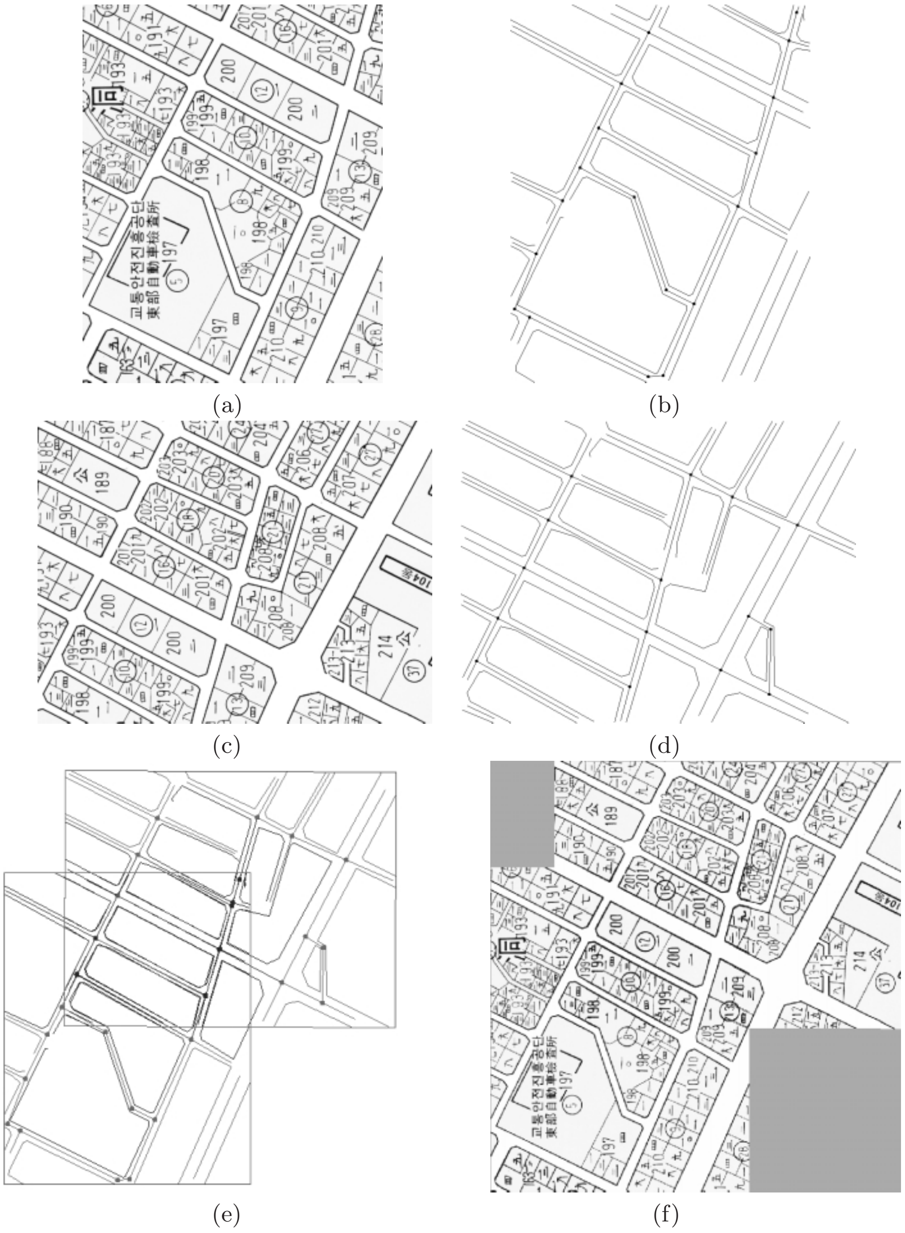

Fig. 8. Examples of the experiment : Problem 3