Geoinformatica (2008) 12:377–410
DOI 10.1007/s10707-007-0033-0

Automatically and Accurately Conflating Raster Maps with Orthoimagery

Ching-Chien Chen · Craig A. Knoblock · Cyrus Shahabi

Received: 20 April 2006 / Revised: 15 June 2007 /
Accepted: 9 July 2007 / Published online: 4 October 2007
© Springer Science + Business Media, LLC 2007

Abstract Recent growth of geospatial information online has made it possible to access various maps and orthoimagery. Conflating these maps and imagery can create images that combine the visual appeal of imagery with the attribution information from maps. The existing systems require human intervention to conflate maps with imagery. We present a novel approach that utilizes vector datasets as "glue" to automatically conflate street maps with imagery. First, our approach extracts road intersections from imagery and maps as control points. Then, it aligns the two point sets by computing the matched point pattern. Finally, it aligns maps with imagery based on the matched pattern. The experiments show that our approach can conflate various maps with imagery, such that in our experiments on TIGER-maps covering part of St. Louis county, MO, 85.2% of the conflated map roads are within 10.8 m from the actual roads compared to 51.7% for the original and georeferenced TIGER-map roads.

Keywords conflation · orthoimagery · street raster maps · vector data · point pattern matching · rubber sheeting

This work is based on an earlier work: Automatically and accurately conflating orthoimagery and street maps, in the Proceedings of the 12th Annual ACM International Symposium on Advances in Geographic Information Systems, {2004} © ACM, 2004. http://doi.acm.org/10.1145/1032222.1032231.

C.-C. Chen (✉)
Geosemble Technologies, 2041 Rosecrans Ave. Suite 245, El Segundo, CA 90245, USA
e-mail: jchen@geosemble.com

C. A. Knoblock
Information Sciences Institute, University of Southern California, Marina del Rey, CA 90292, USA
e-mail: knoblock@isi.edu

C. Shahabi
Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA
e-mail: shahabi@usc.edu

1 Introduction

There is a wide variety of geospatial data available on the Internet, including a number of data sources that provide imagery and maps of various regions. The National Map,[1] ESRI Map Service,[2] MapQuest,[3] Google Map Service,[4] and Microsoft TerraService[5] are good examples of map or imagery repositories. In addition, a wide variety of maps are available from various government agencies, such as property survey maps and maps of oil wells. Satellite imagery and aerial photography have been utilized for real estate listings, military intelligence applications, and other applications. Road vector data covering all of the United States is available from the U.S. Census Bureau.[6] A general problem in combining geospatial data from different sources is that they rarely align. There are a variety of reasons for this problem, but the most common one is that the latest products are collected with higher accuracy and improved processing techniques.

In this paper, we consider the alignment of street maps (i.e., maps showing roads) with orthoimagery (this imagery is altered from original photos so that it has the geometric properties of a map). The system with the capability to align maps and images enables analysts to view an image for any place in the world and then overlay the aligned map to better understand the context of the image they are viewing. We focus on high resolution imagery and maps. This is because it is difficult to view various spatial objects in low resolution imagery (e.g., resolution lower than 16 meters/pixel). So, it is not practical to conflate a map with images of low resolution. Figure 1 shows an example of aligning a street map with a 1 meter/pixel orthoimage. The map is made semi-transparent with the underlying image. This integration can annotate objects on imagery, such as roads, streets and parks, with detailed attribution information contained in diverse maps.

Conflation is often a term used to describe the alignment of different geospatial datasets. The conflation process requires identifying an appropriate set of counterpart features (termed control points) on the two data sources to be integrated and other points will be moved according to the correspondence between the control point pairs [26]. Various GIS researchers and computer vision researchers have shown that the intersection points on the road networks provide an accurate set of control point pairs for diverse geospatial datasets [11], [16], [18], [20]. Consider the conflation of maps with imagery. Currently, the identification of these control points to align maps and imagery is often performed manually, which is a tedious and time-consuming process. Moreover, due to the fact that the coordinates of many online maps are unknown, manually identifying a set of control points from a non-georeferenced map and an image covering a large area is impractical.

In this paper, we present an approach to automatically identify a set of control point pairs from imagery and maps by combining different sources of information from each of the sources to be integrated. In particular, we utilize common vector datasets as “glue” to integrate imagery with maps. Figure 2 shows our overall approach for conflating maps and imagery. We first identify feature points on imagery by aligning vector data with imagery. Then, we compute the correspondence between the image points and the points extracted

[1] http://seamless.usgs.gov

[2] http://arcweb.esri.com/sc/viewer/index.html

[3] http://www.mapquest.com

[4] http://maps.google.com/

[5] http://terraserver-usa.com/

[6] http://www.census.gov/geo/www/tiger/

Fig. 1 An example of the integration of street maps and imagery. **a** Imagery with the area of interest highlighted. **b** ESRI street map. **c** Imagery with aligned map

from the maps. Now that we have a set of control point pairs for the map and imagery, we can use the rubber sheeting technique described in [26] to align the map with the imagery. The resulting system can automatically infer the coordinates of a non-georeferenced map and align with an image covering the overlapping areas.

The approach described in this paper is based on the preliminary techniques that we proposed in [9]. We enhanced our techniques in several ways: (1) we present an enhanced point pattern matching technique, termed GeoPPM, by exploiting auxiliary information (e.g., map scale, the degree of intersections and the density of these intersections) to more efficiently and accurately find matched point patterns across both datasets, (2) we present a novel evaluation methodology to evaluate our conflation results based on three different metrics, and (3) we perform a detailed evaluation on real-world maps of varying accuracy levels to assess our approach. In addition, the GeoPPM algorithm presented in this paper is a significantly enhanced version (improved mainly by exploiting map scales and localized distributions of points) of the point matching algorithm discussed in our previous work [10].

The remainder of this paper is organized as follows. Section 2 reviews our previous work on automatically detecting road intersection points from the imagery and the map, respectively. Section 3 presents our specialized point pattern matching algorithm for finding the mapping between the layout (with relative distances) of the intersection points on the

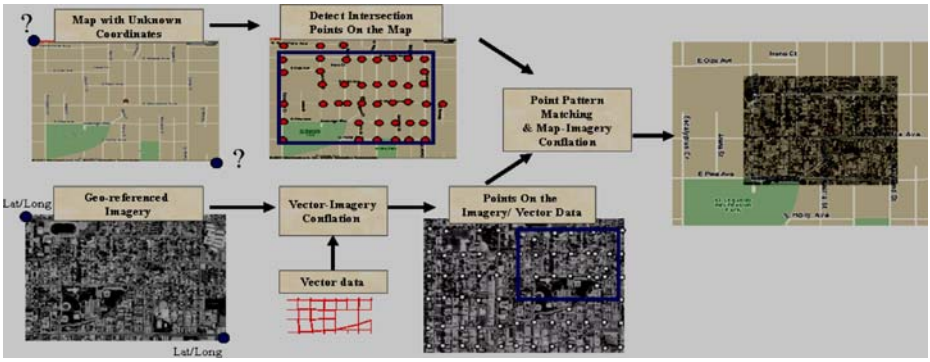


Fig. 2 Overall approach to align orthoimagery and street maps

imagery and the maps to determine the map-coordinate as well as to generate a set of control point pairs. Section 4 describes the approach to aligning maps with imagery based on the detected control point pairs. Section 5 provides experimental results. Section 6 discusses the related work and Section 7 concludes the paper by discussing our future plans.

2 Background

As shown in Fig. 2, detecting prominent features from the map and imagery is a prerequisite step to compare the map and imagery. In this section we review our approaches to detecting feature points (i.e., road intersection points) from imagery and maps, respectively.

2.1 Identifying intersections in imagery

Automatic extraction of road intersection points from imagery is a difficult task due to the complexity that characterizes natural scenes [1]. In order to efficiently and accurately detect road intersection points in imagery, we utilize existing road network vector databases as part of the prior knowledge. In general, integrating existing vector data as part of the spatial object recognition scheme is an effective approach. The vector data represents the existing prior knowledge about the data, thus reducing the uncertainty in identifying the spatial objects, such as road segments and road intersections, in imagery.

In [7], [8], we described the approach for automatic conflation of georeferenced road vector data with georeferenced imagery. We exploit a combination of the knowledge of the road network with image processing in a technique that we call localized template matching [8]. With this approach, we first find road intersection points in the road vector dataset. For each intersection point, we then perform image processing in a localized area around the intersection point to find the corresponding point in the image. The running time for this approach is dramatically lower than traditional image processing techniques due to performing image processing on localized areas. Furthermore, exploiting the road direction and width information improves both the accuracy and efficiency to detect intersections in the image. An issue that arises is that the localized image processing may still identify incorrect intersection points, which introduces noise into the set of control point pairs. To address this issue, we apply an enhanced filtering technique termed the Vector-Median Filter to eliminate inaccurate control point pairs. Once the system has identified an accurate set of control point pairs, we utilize rubber sheeting techniques described in [26] to align the vector data with the imagery.

As show in our test sets described in [7], [8], this approach produces an accurate alignment of the vector data with the imagery. As a result of vector-imagery conflation, the conflated intersection points on the road network are aligned with the intersection points on the imagery. We can then use the conflated intersection points as intersection points on the imagery. Figure 3 shows an example illustrating the detected intersection points on an image before and after conflating the road network with an image.

2.2 Identifying intersection points from street maps

Since the geocoordinates of many online street maps are unknown, we cannot apply the same localized image processing, described in Section 2.1, to find intersection points on maps. This is because we cannot align the vector data with the map since the map geocoordinates are unknown. Hence, in order to deal with a more general scenario, we

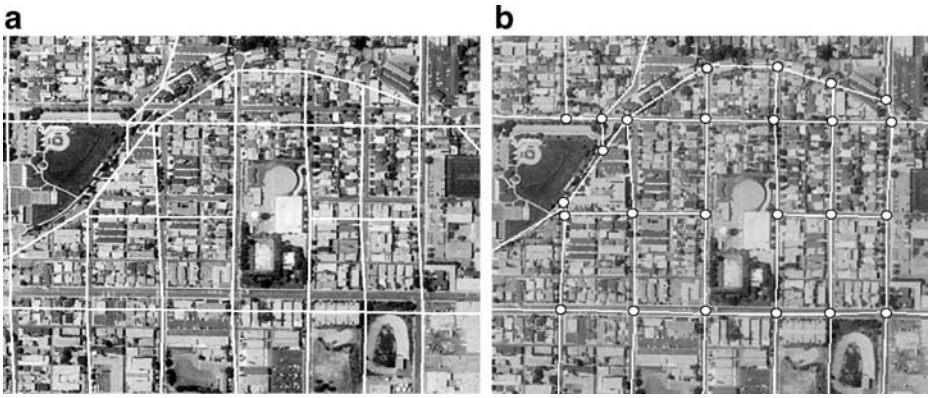


Fig. 3 Intersection points automatically detected on imagery. **a** Imagery with road network, before conflation. **b** Detected intersection points on imagery, after conflation

utilize automatic map processing and pattern recognition algorithms described below to identify the intersection points on maps.

Ideally, intersection points on street maps could be extracted by simply detecting road lines. However, due to the varying thickness of lines on diverse maps, accurate extraction of intersection points from maps is difficult [24], [28]. In addition, there is often noisy information, such as symbols, alphanumeric characters or contour lines on the map, which make it even harder to accurately identify the intersection points. To overcome these problems, we detect road intersections from maps that are preprocessed by a series of image processing techniques. In particular, as discussed in [13], our automated algorithm works as follows: (1) the algorithm analyzes maps to determine the road widths of double line maps in order to more accurately extract potential road segments, (2) it separates the linear structures (e.g., potential roads) from the maps by dynamically investigating thresholds and using the text/graphics separation techniques proposed in [5], (3) it uses morphological operators (including erosion and dilation operators) to reconnect and clarify the potential road segments, and (4) it detects corner points from the remaining lines, and it identifies a point as an intersection point if there are more than two road segments meeting at that point. On average, this algorithm can achieve 95% precision and 75% recall to detect map intersections.[7] Moreover, it has the capability to compute the number of road segments that meet at an intersection (called the degree of an intersection) and the directions of those segments. This additional information can help to improve our point pattern matching algorithm (described next).

Figure 4 shows an example illustrating the detected intersection points on a USGS Topographic Map. Although the algorithm described above can significantly reduce the rate of misidentified intersection points on the maps, it is still possible that noisy points are detected as intersection points or some intersections go undetected. For example, the point near the center of the map (e.g., the detected points in character string "Keysor") was mistaken for a road intersection. However, our point pattern matching algorithm can tolerate the existence of some misidentified intersection points.

[7] A map intersection is characterized as an accurately detected point if and only if its location is less than five pixels from the position of the actual map intersection. For single line map, the actual position of an intersection is the point where the associated road segments meet. For double line map, the points that fall within the polygons formed by the elongated road regions are considered as actual intersections.

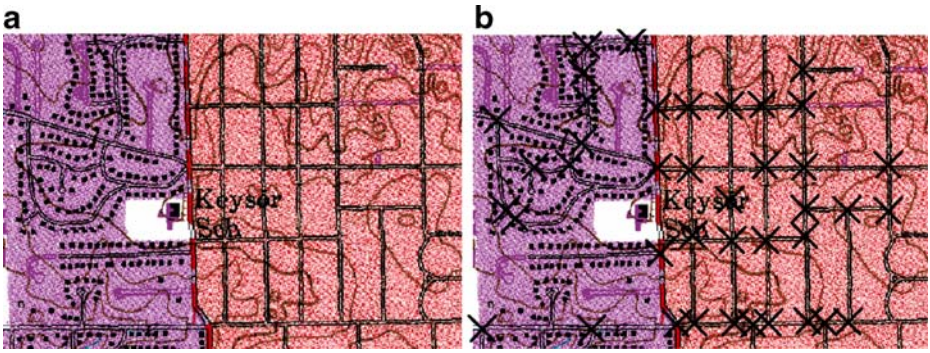


Fig. 4 Intersection points automatically detected on a map. **a** Original topographic map. **b** Intersection points detected on the map (each detected intersection is marked as an X)

3 Generating control point pairs by pattern matching

So far we have identified a set of intersections on both the street map and the imagery. Figure 5 shows an example of the two point sets on a map and an image, respectively. The remaining problem is to find the mapping between these points in order to generate a set of control point pairs.

Let $M = \{m_i \mid m_i = (x_i, y_i)$, where (x_i, y_i) is the location of detected intersections on the map$\}$ and $S = \{s_i \mid s_i = (\text{lon}_i, \text{lat}_i)$, where $(\text{lon}_i, \text{lat}_i)$ is the location of identified intersections on the imagery$\}$. Our objective is to locate the *matched point pair set*: $\text{Rel}_{\text{Pat}} = \{(m_i, s_i) \mid$ where m_i is the accurately detected point on the map and s_i is the corresponding point (if any) on the imagery. That is, the pair m_i and s_i are formed by the same intersected roads.$\}$. Once Rel_{Pat} is identified, the system can use these relevant point pairs to align the map and imagery. Additionally, it can infer the geocoordinates and scale of the map.

To identify Rel_{Pat}, the basic idea is to find the transformation T between the layout (with relative distances) of the intersection point set M on the map and the intersection point set S on the imagery. The key step in matching the two sets of points is the computation of this proper transformation T, which is a 2D rigid motion (rotation and translation) with scaling.

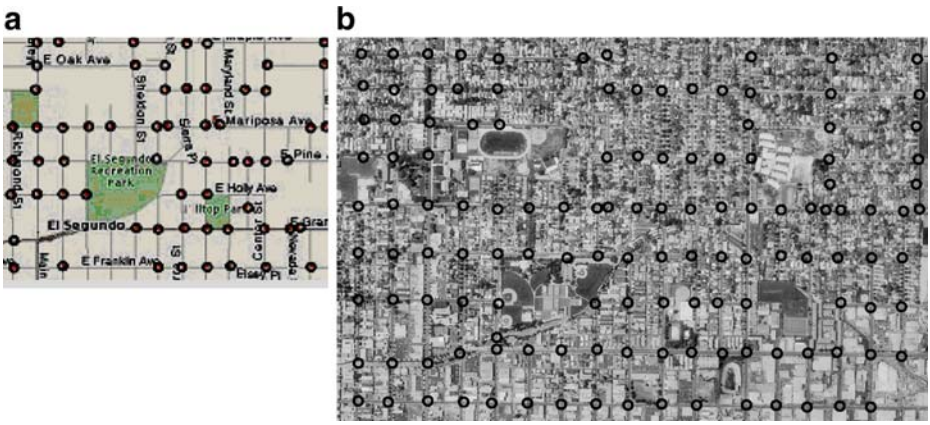


Fig. 5 Intersection points detected on a map and an image. **a** A map with detected intersections. **b** An image with detected intersections

Since most maps and imagery are oriented such that north is up, we only need to compute the translation transformation with scaling. In practice, we compute the transformation when only a fraction α of the points on maps match the points on imagery. The reason that we consider the transformation valid even if only a fraction α of the points on the maps is matched is because there are always misidentified points in the process of identifying intersection points on maps. At the same time, there may be some missing intersection points on the imagery as well. Hence, it is unlikely that 100% of map points match to corresponding points in the imagery. Section 3.1 describes a straightforward algorithm to find the transformation, while Section 3.2 describes our proposed improvement. Table 1 summarizes the notations that we use throughout this section.

3.1 A Naive approach to match point patterns

Our goal is to find the transformation T that matches at least a fraction α of the points of M (on the map) into a subset of S (on the imagery). Symbolically, this implies:

$\exists\ T$ and $M' \subseteq M$, such that $T(M') \subseteq S$, where $\mid M' \mid \geq \alpha \mid M \mid$ and $T(M')$ denotes the set of points that results from applying T to the points of M'. Or equivalently, for a 2D point (x, y) in the point set $M' \subseteq M$, $\exists\ T$ in the matrix form $\begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ Tx & Ty & 1 \end{bmatrix}$ (S_x and S_y are scale factors along x and y direction, respectively, while T_x and T_y are translation factors along x and y directions, respectively), such that for map location x, y:

$[x, y, 1] * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ Tx & Ty & 1 \end{bmatrix}$ = [longitude, latitude, 1] , where $\mid M' \mid \geq \alpha \mid M \mid$ and the 2D point (longitude, latitude) belongs to the intersection point set S on the imagery. Here, we do not expect point coordinates to match exactly because of finite-precision computation or small errors in the datasets. Therefore, when checking whether a 2D point p belongs to the point set S, we declare that p a member of S, if there exists a point in S that is within Euclidean distance δ of p for a small fixed positive constant δ, which controls the degree of accuracy. The minimum δ such that there is a match for M' in S is called the *Hausdorff distance*. Different computations of the minimum *Hausdorff distance* have been studied in great depth in the computational geometry literature [12]. We do not seek to minimize δ but rather adopt an acceptable threshold for δ. The threshold is relatively small compared to the average inter-point distances in S. In fact, this sort of problem was categorized as a "Nearly Exact" point matching problem in [6].

Given the parameters α and δ, to obtain a proper transformation T, we need to compute the values of the four unknown parameters S_x, S_y, T_x and T_y. This implies that at least four different equations are required and the accurate identification of at least two matched point

Table 1 Summary of notations

Symbol	Meaning
M	The set of detected intersections on the map
$\mid M \mid$	Number of detected map intersections = number of items in M
S	The set of identified intersections on the imagery
$\mid S \mid$	Number of identified imagery intersections = number of items in S
T	The computed transformation that transforms some points from one point set to the corresponding points on the other point set
δ	A distance threshold used to determine whether a transformed map point (image point) is matched to an image point (map point).
α	A pre-defined fraction threshold used to define the smallest percentage of map points that must be mapped to image points when applying the computed transformation T

Table 2 Pseudocode of a brute force algorithm to locate matched point pattern

```
Function BruteForcePPM (MapPointSet M, ImagePointSet S)
1.  for each pair m1, m2 ∈ M
2.    for each pair s1, s2 ∈ S                          //pair-generation phase
3.      Compute the transformation T' mapping m1 → s1, m2 → s2, if one exists
4.      Compute T'(M)                                   //apply T' to all map points
5.      if (more than α% of the points in T'(M) match points in S)   //transformation-exam phase
6.          Save the transformation T' and point pair ((m1, s1), (m2, s2))
7.    end for
8.  end for
9.  T = Pick the best stored transformation (based on how many map points could be transformed to
        image points)
10. return T
```

pairs could resolve these unknown parameters. A straightforward (brute-force) method (as shown in the pseudocode in Table 2) to compute transformation T is comprised of two phases:

- *Pair generation phase* is to produce all possible matched point pairs by exhaustively permuting and mapping the points from the map to the imagery.
- *Transformation examination phase* is to utilize each of these potential point pairs to find its corresponding transformation. Among all these transformations, we then identify those transformations that correlate at least a fraction α of the map points to the image points as potential candidates.

More precisely, this algorithm first chooses a point pair (x_1, y_1) and (x_2, y_2) from M, then, for every pair of distinct points $(\text{lon}_1, \text{lat}_1)$ and $(\text{lon}_2, \text{lat}_2)$ in S, the transformation T' that maps the point pair in M to the point pair in S is computed by solving the following four equations:

$$Sx * x_1 + Tx = \text{lon}_1 \quad Sy * y_1 + Ty = \text{lat}_1$$
$$Sx * x_2 + Tx = \text{lon}_2 \quad Sy * y_2 + Ty = \text{lat}_2$$

Each generated transformation T' is thus applied to the entire set of points in M to check whether there are more than $\alpha |M|$ points that can be aligned with some points in S within the threshold δ. This process is repeated for each possible point pair from M, which implies that it could require examining $O(|M|^2)$ pairs in the worst case. Since for each such pair, the algorithm needs to try all possible point pairs on S (i.e., $O(|S|^2)$) and spends $O(|M| \log |S|)$ time to examine the generated transformation T', this method has a worst case running time of $O(|M|^3 |S|^2 \log |S|)$. The advantage of this approach is that we can find a mapping (if the mapping exists) with a proper threshold δ, even in the presence of very noisy data. However, it suffers from a high computation time. One way to improve the efficiency of the algorithm is to utilize randomization in choosing the pair of points from M as proposed in [23], thus achieving the running time of $O(|S|^2 |M| \log |S|)$. However, the approach by Irani et al. is not appropriate for our datasets because the extracted intersection points from maps and imagery could include a number of misidentified intersection points. In addition, there could be some missing intersections on both point sets. Instead, we present an efficient technique in the following section to improve the computation time of this algorithm.

3.2 Enhanced point pattern matching algorithm: GeoPPM

We improve the brute-force point pattern matching algorithm by reducing the number of potential matching point pairs that need to be examined. The basic idea is to exclude all unlikely matching point pairs during the *pair generation phase* by exploiting auxiliary information, such as map scale (or map resolution[8]), the degree of an intersection (i.e., the number of intersected road segments), and the density of these intersections. For example, given a point pair (x_1, y_1) and (x_2, y_2) of M, we only need to consider those pairs $(\text{lon}_1, \text{lat}_1)$ and $(\text{lon}_2, \text{lat}_2)$ of S, such that the real world distance between (x_1, y_1) and (x_2, y_2) is close to the real world distance between $(\text{lon}_1, \text{lat}_1)$ and $(\text{lon}_2, \text{lat}_2)$. In addition, (x_1, y_1) and $(\text{lon}_1, \text{lat}_1)$ would be considered as a possible matching point if and only if they have similar road degrees and road directions.

Since the geometric point set matching in two or higher dimensions is a well-studied family of problems with application to different areas such as computer vision, biology, and astronomy [12], [23], we do not intend to invent a novel algorithm to resolve the general point pattern matching problem. Instead, we focus on the datasets we are conflating and particularly design efficient and accurate matching algorithms to discover geospatial point patterns. In particular, we consider the alignment of the map and imagery of similar resolution. As stated in Section 1, high resolution imagery is the target of our system. Meanwhile, most online high resolution maps depict detailed road network distribution (without simplification or generalization). This implies a similar level of detail of road networks in the two datasets, such that we can optimize the matching process by exploiting map scale, similarity of road direction, and similarity of intersection density, etc. The above-mentioned ideas are the core ideas behind our algorithm, termed GeoPPM, implemented in our system. Figure 6 illustrates the GeoPPM algorithm. We will describe it in detail in the following sections.

3.2.1 Improvement by exploiting map scale

If the map scale is provided, we further improve the (brute-force) point matching algorithm by exploiting information on direction and relative distances available from the vector sets and maps. The information on direction and distance is used as prior knowledge to prune the search space of the possible mapping between the two datasets. More precisely, we improve the performance by transforming the points on maps and imagery to a 2D Euclidean space, where the distance measure is ground distance. The real world distance is used between points in the transformed space. Therefore, we only consider translation without scaling in such a space.

In particular, the process of choosing the original point pair (as shown in Fig. 7) can be divided into the following subtasks: (1) choose one point P from the map as the origin (0,0), then determine the coordinates of other points Q_i (X_i, Y_i) as follows. X_i is the ground distance between P and Q_i in east–west orientation, while Y_i is the ground distance between P and Q_i in north–south orientation. Note X_i is negative, if Q_i is west of P. Y_i is negative, if Q_i is south of P. (2) apply the similar transformation to the points on the imagery. We can now compare the two point patterns by computing the translation T between the two transformed point patterns. The revised algorithm improves the time complexity to $O(|M|^2$

[8] We can determine the map resolution for a raster map from the known map scale.

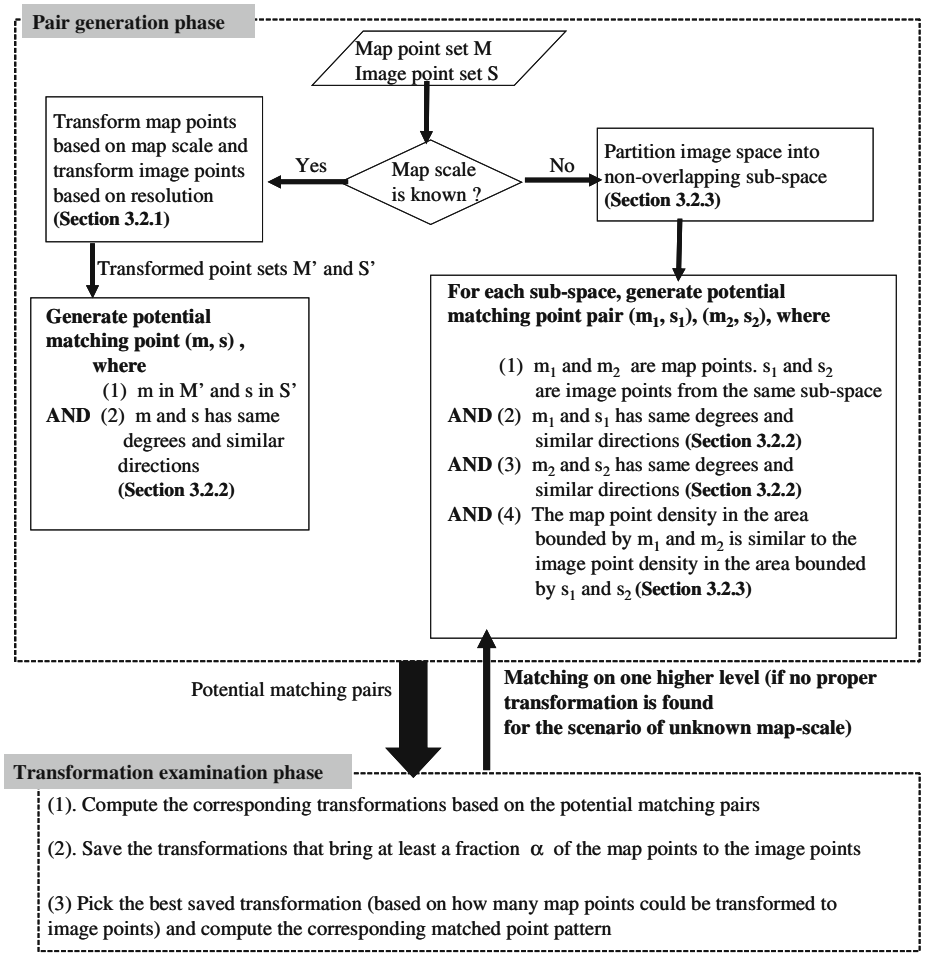


Fig. 6 The GeoPPM algorithm

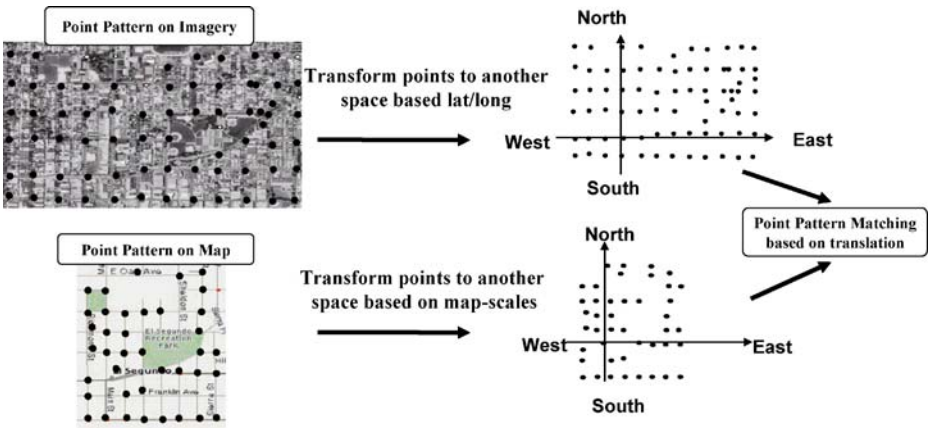


Fig. 7 Enhanced point pattern matching process using map scales

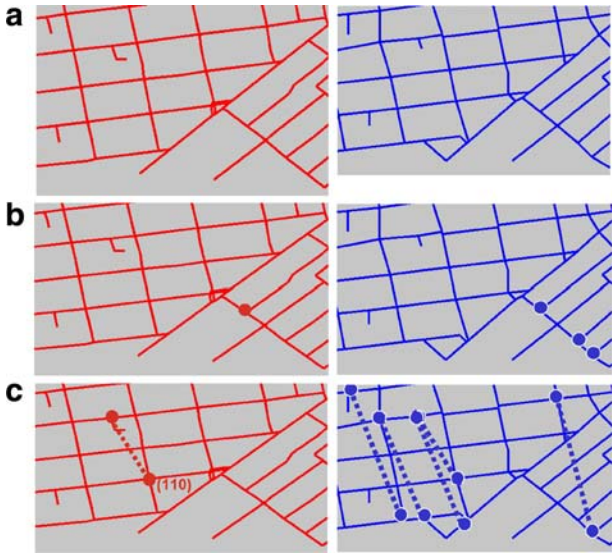


Fig. 8 Comparing two road networks (on the map and imagery) by exploiting geometric information. **a** The two networks (from the map and imagery) to compare. **b** The points highlighted in the right figure have the same *connectivity* and *angles* as the point highlighted in the left figure. **c** The point pairs highlighted in the right figure have the similar *angles* (between the points) as the point pair highlighted in the left figure

$|S|\ log|S|$), due to the fact that we need to consider only the potential *matched points* (instead of *matched point pairs*).

3.2.2 Improvement by exploiting geometric information

The intuition behind GeoPPM is to increase the possibility of examining the correct matching pair from the candidates by first excluding all unlikely matching point pairs. Prominent geometric information associated with the road intersections often can differentiate one point from another point and can be used to exclude a huge amount of potential matching point pairs. More specifically, in addition to utilizing map scales, we improve the point pair generation process to discover good potential candidates by exploiting the following geometric information:

1. *Point degree*: We define the degree of a point as the number of the road segments that intersect at that point. Clearly, if datasets M and S have very close level-of-detail (e.g., similar number of intersections per one unit of area), a candidate matching point P_1 in M for a point P'_1 in S must have the same connectivity.
2. *Angles of the point*: The angles of a point are defined as the angles of the road segments that intersect at that point. Similar to the point degree, a point P_1 in M can only be considered as a candidate for point P'_1 in S only if the two points have similar angles, or the difference between their angles is less than a threshold value. To illustrate, consider comparing two road networks as the example shown in Fig. 8(a). Whenever the system chooses a point (as the point shown in the left figure of Fig. 8(b)) in one road network, it only has to consider the candidate matched points with same degree and similar directions of intersected road segments from the other network (as some possible candidates marked in the right figure of Fig. 8(b)).
3. *Angle between the points*: The angle between two points is defined as the angle of the straight line that connects the points. Clearly, a pair (P'_1,P'_2) can be considered as a possible candidate for the pair (P_1, P_2) only if the angle between P'_1 and P'_2 is similar to the angle between P_1 and P_2, or the difference between their angles is less than a

threshold value. Consider the example shown in Fig. 8(c). Whenever the system chooses a point pair (as the point pair shown in the left figure of Fig. 8(c) and the angle between these two points is about 110°) in one road network, it only has to consider the candidate matched point pairs with the similar angle (as some possible candidate point pairs marked as dash lines in the right figure of Fig. 8(c)).

3.2.3 Improvement by exploiting point density and the localized distribution of points

The point distribution information can help to exclude unlikely matching point pairs as well, because, intuitively, the point distributions are often varying in different localized areas. Hence, GeoPPM investigates the point distribution (and density) in localized areas and combines with the geometric information exploited at each point as described in Section 3.2.2 to rapidly discover potential matching point pairs. More precisely, without loss of generality, we consider the scenario that the point set on the map is a subset of the point set on the imagery and the map scales are unknown in advance. We in turn describe the localized point distribution information exploited in GeoPPM:

1. *Point density*: The density of the points in the map should be similar to the density of the matched points in the imagery. As an example shown in Fig. 9, given a point pair P_1 and P_2 of M, we do not need to consider pairs Q_1 and Q_2 of S. This is because the number of points (about 40 points in this example) included in the bounding box Bm (formed by P_1 and P_2) is significantly different from the number of points (about 800 points) in the bounding box Bs (formed by Q_1 and Q_2).
2. *Localized distribution of points*: The points of the matched pattern tend to scatter in neighboring (or localized) areas. Hence, it is not necessary to evaluate the whole search space in one step but it is sufficient to partition the search space into smaller sub-parts and evaluate each independently (i.e., the desired transformation can be computed from some potential matching point pairs locally without considering all pairs from the entire data set). Consider the example shown in Fig. 10. There are 57 detected intersections on the map and there are 1,059 intersections on the image.[9] The image space is partitioned into 16 equi-sized cells (e.g., cells AFQB, BQTC, etc). In order to explain our algorithm, the matched point pattern on the imagery is highlighted by a dashed rectangle and we also mark some matching point pairs (e.g., the points m_1, m_2 ,m_3, m_4, m_5 and m_6 on the maps correspond to the six points s_1, s_2, s_3, s_4, s_5 and s_6 shown in the enlarged dashed area of Fig. 10(c)).

As shown in the example of Fig. 10, the system first chooses a point pair $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ from M. Then, for every pair of distinct points $Q_1(\text{lon}_1,\text{lat}_1)$ and Q_2 $(\text{lon}_2,\text{lat}_2)$ in the same cell (e.g., cell AFQB), the system computes the transformation T, if

- There is similar point density in the bounding boxes formed by (P_1, P_2) and (Q_1, Q_2), respectively.
- The road directions of P_1 (P_2) are similar to the road direction of Q_1 (Q_2).

[9] We removed the background imagery and road networks in order to clearly display the distribution of points on the imagery.

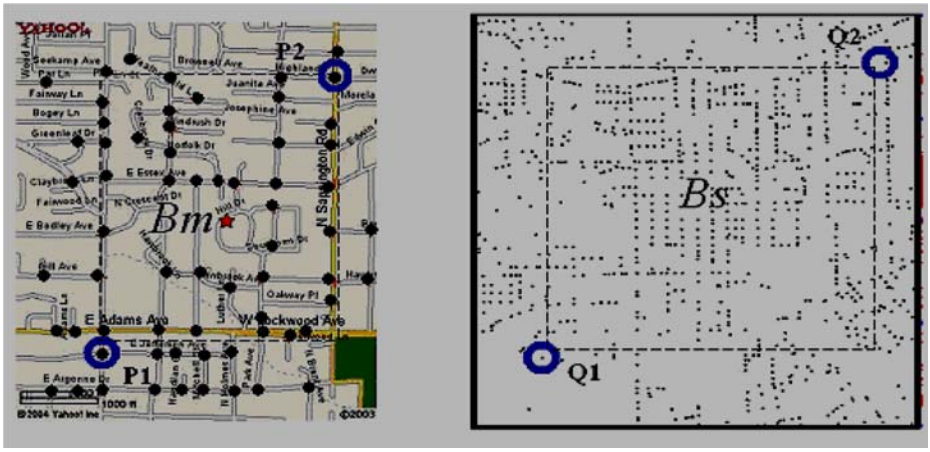


Fig. 9 An example of utilizing point density to prune the search space

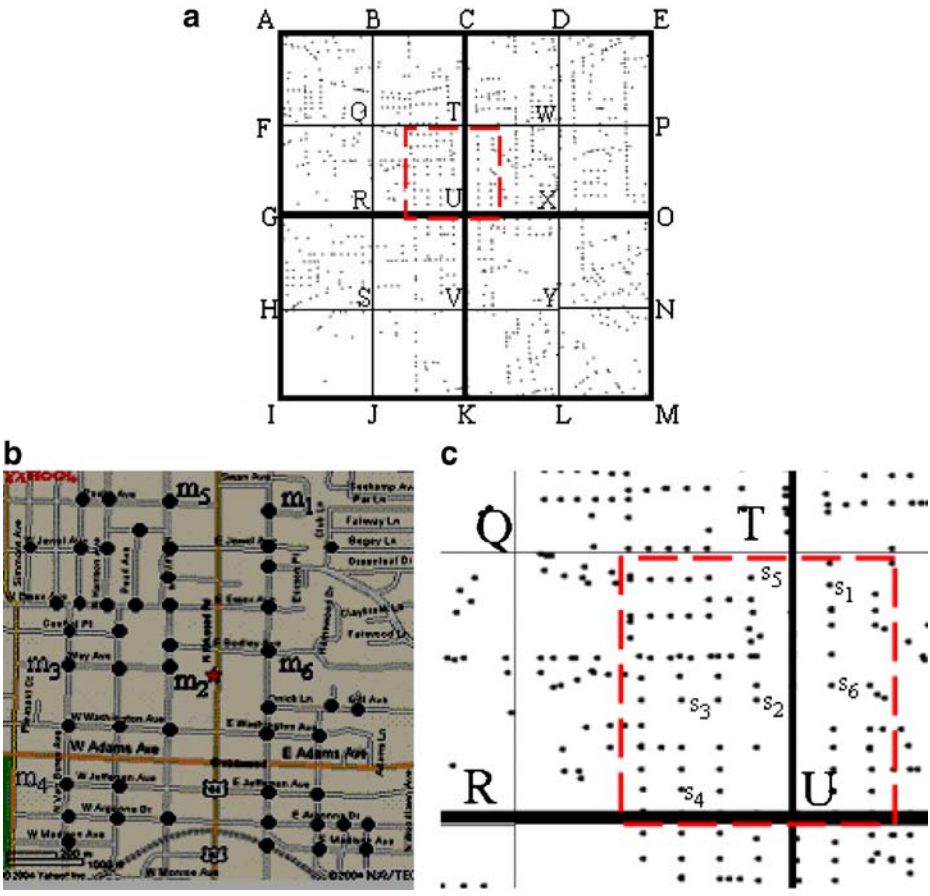


Fig. 10 An example of utilizing localized point distribution to prune search space. **a** Partition the image space (with 1059 intersections) into 16 equi-sized cells. The corresponding area to the map is highlighted. **b** 57 detected intersections on a map. **c** The enlarged overlapping area in the imagery

The system then applies transformation T to the entire points in M to check whether there are more than $\alpha|M|$ points that can be matched with some points on S within the threshold δ. The above-mentioned process is repeated for each possible point pair from M.

With proper settings of α and δ, it is very likely that the system can obtain some correct matching point pairs, such as $((m_2, s_2), (m_5, s_5))$[10] or $((m_3, s_3),(m_4, s_4))$ in cell QRUT or $((m_1,s_1),(m_6, s_6))$ in cell TUXW. This saves running time, because the approach does not need to examine the point pairs that are located in different cells (e.g., the image points s_1 and s_4). The failure of identifying matched pattern in lower level cells will cause the algorithm to search the pattern in higher levels. However, since the points of the matched pattern tend to scatter in localized areas and the two datasets (i.e., the map and imagery) are in the similar level-of-detail, there is a higher possibility that our approach can find some matching point pairs in the cells of lower levels.

We developed a hierarchical grid structure (called HiGrid) to implement the above-mentioned idea. An example of HiGrid where the system recursively subdivides the space into four sub-spaces to the depth 3 is shown in Fig. 11. The depth k (the highest level has the depth zero) of HiGrid is calculated (during the construction of HiGrid) based on the number of points in the imagery. Assume that there are $|S|$ points in the image and the system partitions the grid into b sub-grids when building the HiGrid structure. In addition, assume that the points are uniformly distributed over the space and we intend to have n points (on average) for each cell of the lowest level. Hence, we can infer the inequality:

$$n^* b^k \leq |S| < n^* b^{k+1} \tag{1}$$

This implies that the depth k of the HiGrid structure is $\lfloor \log_b \frac{|S|}{n} \rfloor$.

Utilizing HiGrid results in an efficient, systematic and hierarchical way to search for matched point patterns in local (i.e., small) regions. Furthermore, each cell of the same level is independent (i.e., there is no overlaps) and can be processed in parallel.

3.2.4 Summary

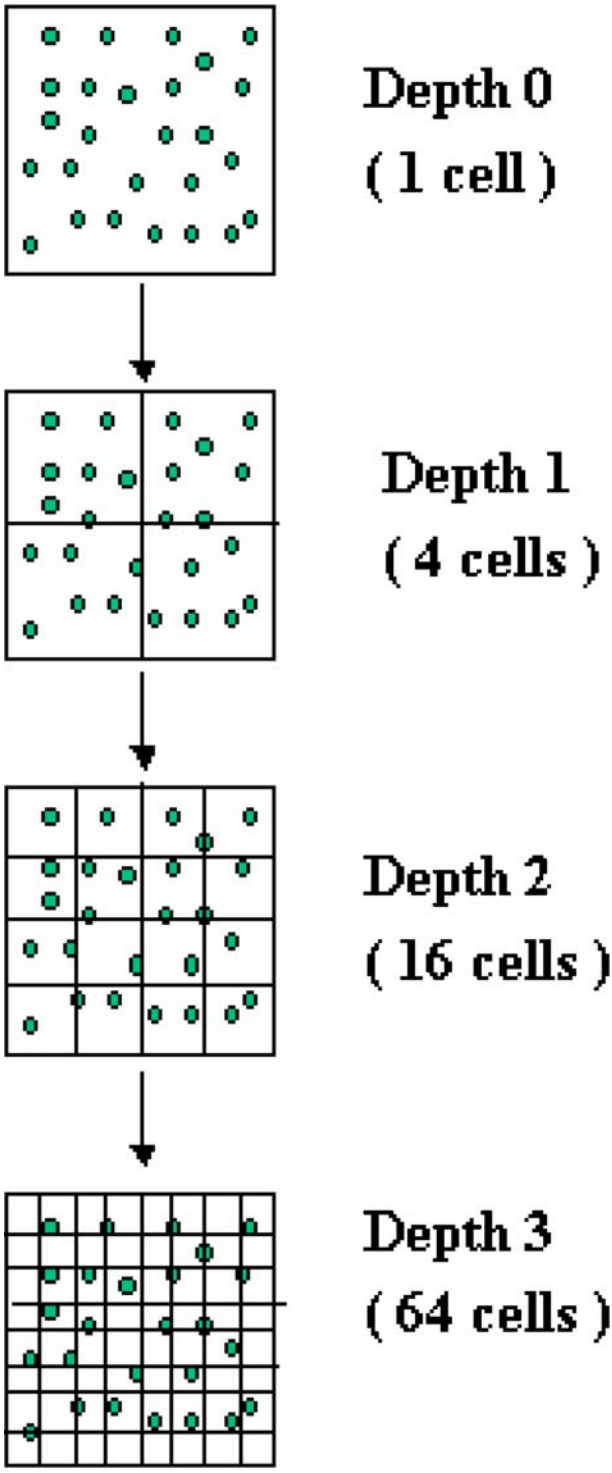
In sum, our GeoPPM approach exploits geospatial information (e.g., map scale, road directions and density of intersections) to prune search space. More precisely, if the map scale is known in advance, GeoPPM utilizes map scale and road intersection directions to identify the common point pattern. Otherwise, it identifies the point pattern by using the HiGrid structure, point density and road intersection directions. In our current implementation of GeoPPM, we do not use HiGrid for maps with known map scales (due to the fact that we have achieved acceptable performance by just using the map scale, as described in Section 5). In general, however, we can also utilize HiGrid structure and point density for maps with known map scales.

4 Image and map conflation

Now that we have a set of control point pairs for the map and imagery (as in the example shown in Fig. 12), we can deform one of the datasets (the source image) to align the other (the target image) utilizing the identified control point pairs. Without loss of

[10] The matching point pairs notation $((m_i, s_i),(m_j, s_j))$ implies that m_i matches s_i and m_j matches s_j.

Fig. 11 An example of HiGrid



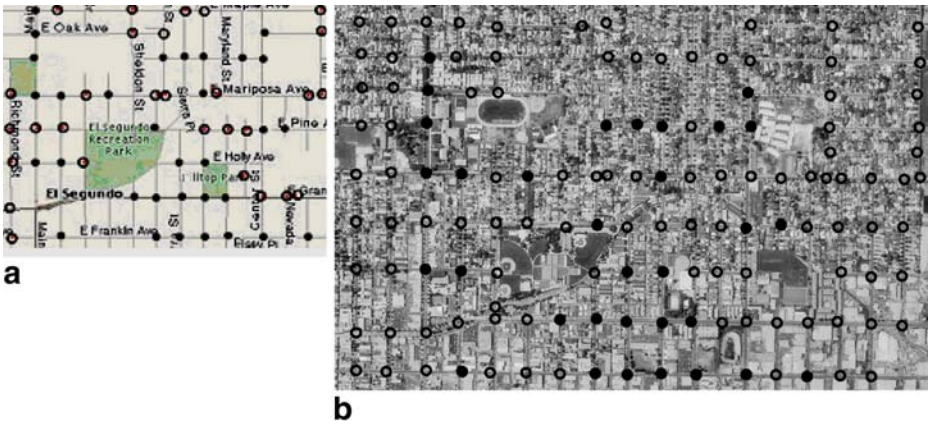


Fig. 12 A sample result of GeoPPM. **a** A map with matched point pattern highlighted with black dots. **b** An image with corresponding point pattern highlighted as black dots

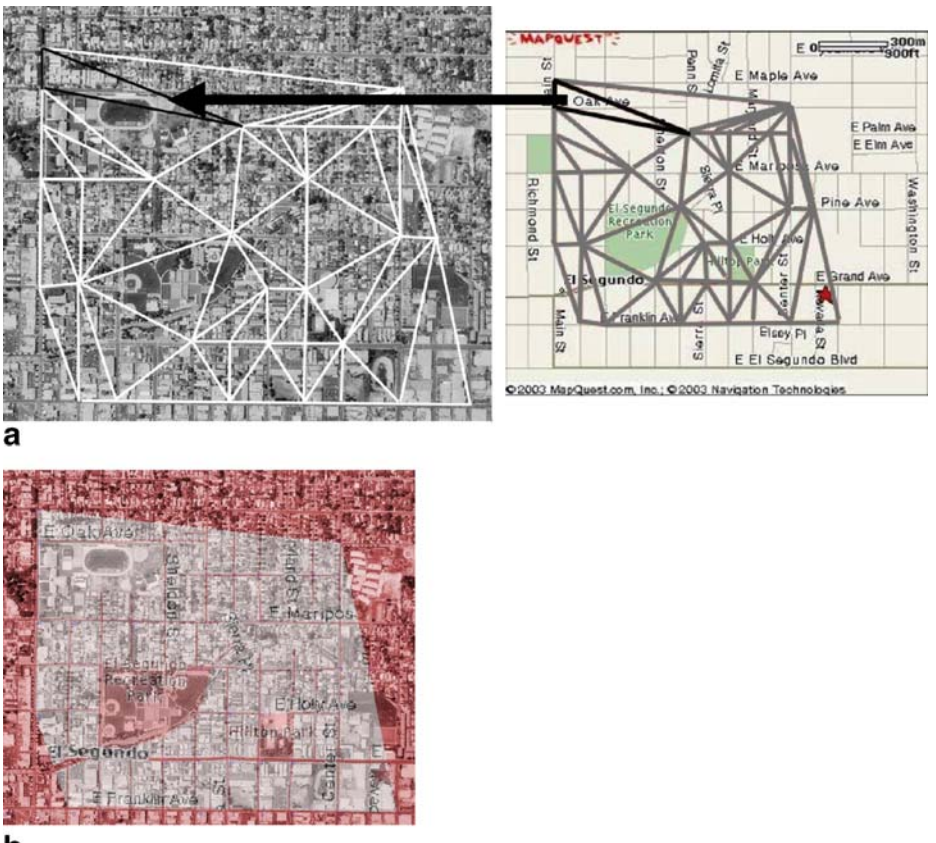


Fig. 13 Delaunay triangulation and rubber sheeting. **a** Delaunay triangulation on the imagery and the map. **b** Conflation result after applying rubber sheeting

Table 3 Datasets used in experiments

Dataset	Test Set 1 (El Segundo, CA)	Test Set 2 (St. Louis, MO)
Imagery	Georeferenced USGS DOQ orthoimagery with 1 meter/pixel resolutions	Georeferenced USGS high resolution color orthoimagery with 0.3 meter/pixel resolution
Maps (with various sizes and scales)	5 ESRI maps 5 MapQuest maps 5 Yahoo maps 5 TIGER maps 5 USGS Topographic maps	5 ESRI maps 5 MapQuest maps 5 Yahoo maps 5 TIGER maps 5 USGS Topographic maps
Vector data	U.S. Census TIGER/Lines Length: 84.32 km About 300 intersections	USGS MO-DOT road vector Length: 364.28 km About 1130 intersections
Area covered	Latitude: 33.9164 to 33.9301 Longitude: −118.4351 to −118.3702 Width: 5.2 km Height: 1.6 km	Latitude: 38.5534 to 38.6091 Longitude: −90.4389 to −90.3720 Width: 6 km Height: 6 km

generality, we assume that the map is the source image, while the orthoimage is the target image.

To achieve overall alignment of an image and a map, the system must locally adjust the map to conform to the image. We align the two datasets based on local adjustments since small changes in one area should not affect the geometry at large distances. To accomplish local adjustments, the system partitions the domain space into small pieces. Then, we apply local adjustments on each individual piece. Triangulation is an effective strategy to partition the domain space to define local adjustments. There are different triangulations for the control points. One particular triangulation, the Delaunay triangulation, is especially suited for the purpose of conflation [25], [26]. A Delaunay triangulation is a triangulation of the point set with the property that no point falls in the interior of the circumcircle of any triangle (the circle passing through the three triangle vertices). The Delaunay triangulation maximizes the minimum angle of all the angles in the triangulation, thus avoiding triangles with extremely small angles. We perform the Delaunay triangulation with the set of control points on the map and make a set of equivalent triangles with corresponding control points on the imagery. The details of the triangulation algorithms can be found in [4], [22].

Next, the system deforms the map algorithmically, forcing registration of control points on the map with their corresponding points on the imagery. This technique is called "Rubber sheeting" [31]. There are two steps to rubber sheeting. First, we calculate the transformation coefficients to match each Delaunay triangle on the map into its corresponding triangle on the imagery. Second, for each pixel in each triangle on the imagery, we replace it semi-transparently with the corresponding pixel on the map by using the computed transformation coefficients.

Figure 13(a) shows an example of Delaunay triangulation, and the arrow illustrates that the pixels of the triangle on the imagery would be (semi-transparently) overlaid by the corresponding pixels on the map (i.e., rubber sheeting). In practice, if the conflation area (i.e., the convex hull formed by control points) of the source image is much larger than that of the target image, the rubber sheeting results will be distorted because the sampling

Fig. 14 Sample imagery in Test Set 2

frequency is insufficient. We solve this problem by rescaling the conflation area on the map and imagery to identical sizes before applying triangulation and rubber sheeting. Figure 13(b) shows the result after applying the rubber sheeting technique based on the Delaunary triangulation.

5 Performance evaluation

We utilized a set of online street maps and imagery to evaluate our approach. The purpose of the integration experiment was to evaluate the utility of our algorithms in integrating real world data. We are interested in measuring the accuracy of the integration of maps and imagery using our techniques. To that end, we performed several experiments to validate the hypothesis that using our techniques we can automatically and accurately align maps and imagery.

Section 5.1 describes the experimental setup and the test datasets. Section 5.2 presents our evaluation methodology to measure the performance. Section 5.3 discusses the experimental results.

5.1 Experimental setup

Table 3 summarizes the datasets and test sites used for our experiments. We describe these datasets in turn.

(1) Orthoimagery

The imagery used in the experiments is georeferenced USGS high resolution color orthoimagery (with 0.3 m per pixel resolution) and georeferenced USGS gray-scale DOQ imagery with 1 meter/pixel resolution. In particular, we tested color imagery that covers an area of the county of St. Louis, MO, and gray-scale imagery that covers an area of the city of El Segundo, CA. This imagery is available online and can be

queried from the Microsoft TerraService Web Service [3]. Figure 14 shows about 0.6% of the imagery used in Test Set 2.

(2) Street maps

We used several streets maps queried from five different online map services. They are ESRI map,[11] MapQuest map,[12] Yahoo map,[13] U.S. Census TIGER map and USGS topographic map.[14] Although these street maps mainly depict roads, sometimes they also show and name prominent natural and cultural features, such as rivers, parks and schools. The details of these maps are as follows.

- ESRI maps are generated based on data from Geographic Data Technology (GDT). They are high quality street map data with highly accurate street geometry. Neither map scale nor geocoordinates for the ESRI maps are provided online.
- MapQuest maps and Yahoo maps are produced based on NAVTEQ NAVSTREETS. They are also high quality street map data with highly accurate street geometry and they illustrate street maps in diverse map scales, sizes and colors.
- U.S. Census TIGER maps are generated from TIGER/Line files. The TIGER system was developed by the U.S. Bureau of Census. The Census Bureau has developed the TIGER/Line files, which are extractions of selected geographic and cartographic information (including roads) from the TIGER database. TIGER maps have poor positional accuracy and poor road geometry.
- USGS topographic maps depict roads, contour lines to show elevation differences and prominent natural and cultural features of an area. Such detail is useful for local area planning and helpful to hikers (because this map can show elevation changes along a trail).

Note that only TIGER-maps and USGS topographic maps are provided with geographic coordinates. For the purpose of our experiments, this information is ignored. The maps evaluated in these experiments involve various map resolutions (or map scales) ranging from 1.2 to 14.08 m/pixel. Figure 15 provides examples of these street maps.

(3) Vector data (road networks)

Two different road networks were used as "glue" to align maps with imagery: U.S. Census TIGER/Lines is utilized in Test Set 1, while USGS MO-DOT data[15] is used in Test Set 2. Figure 16 shows samples of the road networks utilized.

Our automatic map-imagery conflation system was developed in C#. The algorithm allows the user to specify the two datasets to conflate. The output of our conflation system is an image showing the alignment of the map and imagery. Also note that the thresholds δ and α, used in the point pattern matching routine, were determined experimentally. In

[11] ESRI provides various online map services. In order to evaluate our proposed map-imagery conflation technique for maps with unknown map scale, we used the ESRI maps available at http://arcweb.esri.com/sc/viewer/index.html in the experiments. Neither map scale nor geocoordinates of ERSI maps are provided from this web site.

[12] http://www.mapquest.com

[13] http://maps.yahoo.com/

[14] http://terraserver-usa.com/

[15] MO-DOT is the road network data provided by the Missouri Department of Transportation. It is high quality vector data with highly accurate road geometry.

Fig. 15 Samples of different street maps used in the experiment. **a** An ESRI map with unknown map scale. **b** A MapQuest map with resolution 4.8 meters/pixel. **c** A TIGER map with resolution 4.17 meters/pixel. **d** A topographic map with resolution 2 meters/pixel

particular, we randomly selected three maps from the test datasets and then tried to gain the best matching results by tuning the value of α. The α value we investigated for this experiment was 70% and the δ value was 30 meters.

We conducted the following experiments. We first obtained online orthoimages covering the experimental areas and identified road intersection points on the images by utilizing

Fig. 16 The samples of road networks used in experiments. **a** TIGER/line road network used in test area 1. **b** MO-DOT road network used in test area 2

information inferred from the vector dataset (as described in Section 2.1). Then, we randomly downloaded various street maps (with diverse sizes and map scales) within these areas from the above-mentioned five map services and extracted an intersection point set for each map. Then, GeoPPM computed the alignment between the point set on each map with the point set on the image. Finally, the system aligned each map and imagery based on the matched point pattern. We evaluated the conflation results by the evaluation methodology discussed in Section 5.1 and present the results in Section 5.2.

5.2 Evaluation methodology

Since the evaluation of map-imagery conflation has not been studied before, we developed a novel evaluation method to measure how well the features on the map align to the corresponding features on the imagery. In particular, we consider how well the conflated map roads align to the corresponding roads on the imagery. Towards this end, we "vectorize" the conflated map road pixels, partition the roads into smaller road segments, and utilize the similar evaluation schema (called road-buffer method) described in our previous work on vector-imagery conflation [8] to compare the conflated (and vectorized) map road network with the real imagery road network, termed reference road network. The reference road network is composed of manually plotted road axes (segments) and road sides that represent the ground truth.

Since the accuracy of the matched points significantly affects the conflation results, we describe the measurements to assess the performance of GeoPPM in Section 5.2.1. We then present the detailed methodology to evaluate the conflated map roads in Section 5.2.2.

5.2.1 Evaluation methodology to assess the performance of GeoPPM

We present two metrics, precision and recall, to measure the performance of GeoPPM. The point pattern generated by GeoPPM can be defined as a set:

$\text{Ret}_{\text{Pat}}=\{(m_i, s_i) \mid$ where m_i is the point on the map and s_i is the corresponding imagery point returned by GeoPPM$\}$.

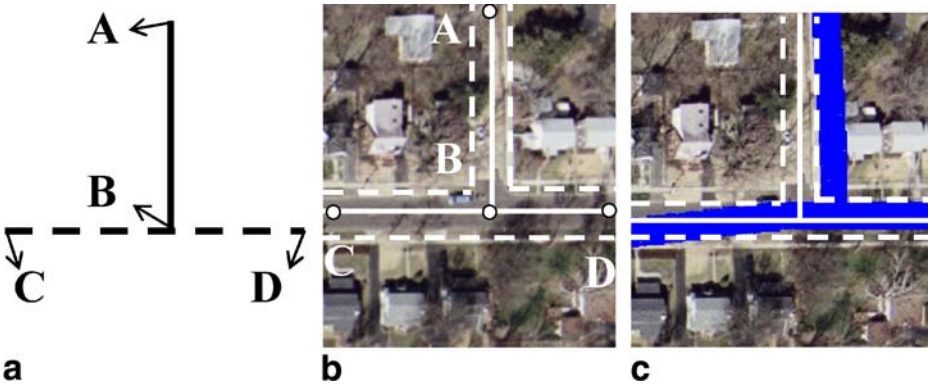


Fig. 17 Map–imagery conflation evaluation by "road buffer" method. **a** Layout of the reference roads (two road segments AB and CD represent the real roads). **b** The overlay of reference roads with imagery (dashed lines: the road sides that form the buffer around the reference roads). **c** Conflated map roads to be evaluated (solid regions: conflated map roads; white lines: reference roads; dash lines: road sides)

To measure the performance of GeoPPM, we need to compare the set Ret_{Pat} with respect to the relevant matched point pattern set Rel_{Pat} (defined in Section 3).

Using these terms, we define precision and recall as follows:

$$\text{Precision} = \frac{\left|\text{Ret}_{\text{pat}} \cap \text{Rel}_{\text{pat}}\right|}{\left|\text{Ret}_{\text{pat}}\right|} \tag{2}$$

$$\text{Recall} = \frac{\left|\text{Ret}_{\text{pat}} \cap \text{Rel}_{\text{pat}}\right|}{\left|\text{Rel}_{\text{pat}}\right|} \tag{3}$$

Intuitively, precision is the percentage of correctly matched point pairs with respect to the total point pairs detected by GeoPPM. Recall is the percentage of the correctly matched point pairs with respect to the actual matched point pairs. In our experiments, the set Ret_{Pat} qualifies as a matched point pattern if and only if precision is greater than 80% and recall is higher than 70%. We tolerate lower recall because the conflation process does not require a large number of control point pairs to perform accurate alignment. In fact, a smaller set of control points with higher accuracy would be sufficient for the conflation process, and hence the higher precision requirement.

5.2.2 Evaluation methodology to assess the conflated map road network

Figure 17 shows an example that we will use through out this section. In Fig. 17(a), the road segments AB and CD are reference road axes. In Fig. 17(b), the dash lines represent the road sides and they form the "buffer" around the road axes. In Fig. 17(c), the solid regions are conflated map roads to be evaluated against the real roads segments AB and CD.

We define the following terms:

- RR is the set of road segments that compose the reference road network.
- CR is the set of road segments that compose the conflated map road network.
- MRR is the set of reference road segments that can be matched to the corresponding conflated map road segments. In other words, for each segment l belonging to MRR, we can find a corresponding segment c in CR, such that l and c have consistent directions and c is within the buffer around l.
- MCR is the set of conflated map road segments that can be matched to the corresponding reference road segments. In other words, for each segment b belonging to MCR, we can find a corresponding segment w in RR, such that b and w have consistent directions and b is within the buffer around w.

Using these terms, two measurements, completeness and correctness, are defined as follows[16]:

$$\text{Completeness} = \frac{\text{Length(MRR)}}{\text{Length(RR)}}$$
$$\text{Correctness} = \frac{\text{Length(MCR)}}{\text{Length(CR)}}$$

[16] Note that if the shapes of corresponding road segments in the set MRR and MCR are different, the lengths of these corresponding road segments may be different.

Intuitively, completeness corresponds to "*the percentage of the reference roads in imagery for which our approach generated conflated map roads"* and it implies "how complete is the conflated map road network (i.e., what's missing in the road network, if we replace the reference road network with the conflated map road network)". Correctness corresponds to "*the percentage of correctly conflated map roads with respect to the total conflated map roads*" and it means "how correct is the conflated map road network".

To illustrate, consider the example shown in Fig. 17. Segments AB and CD belong to the set RR. Segments CD belongs to MRR, since the horizontal conflated map roads (see Fig. 17(c)) match to the reference road segment CD (since these conflated map roads are in the buffer defined by road sides and they have consistent directions with segment CD). Segment AB is not in MRR, since the conflated road does not fall in the buffer. On the other hand, the horizontal conflated map road (as in Fig. 17(c)) belongs to MCR (and CR), because the map road pixels are in the road buffer and have consistent directions with reference segment CD. The vertical conflated map road does not belong to MCR (but belongs to CR).

Additionally, the third measurement metric, *positional accuracy* (based on the technique proposed in [19]) is used to calculate "*the percentage of the total length of the conflated map roads that is within a specific distance of the reference roads*". More precisely, we consider a buffer with width x around the reference road axes, then compute the proportion of the conflated map road pixels that lies within the buffer. In our experiments, we varied x from 3.6 m (i.e., the U.S. standard lane width) to 32.4 m (i.e., nine times the U.S. standard lane width).

5.3 Experimental results

We discuss the performance of GeoPPM algorithm and the performance of the overall map-imagery conflation as follows.

5.3.1 Performance of GeoPPM

After conflating road vector data with imagery, the system identified 281 intersection points on the image of Test Set 1 and 1059 intersections on the image of Test Set 2. Because the tested maps are of diverse sizes and scales, the number of points detected on each map is different. On average, there are about 60 points on each map.

When applying GeoPPM to these detected point sets, our system exploited additional information to improve the performance (as stated in Section 3). In particular, if the map scale is known in advance, the system utilized map scale and road intersection directions to identify the common point pattern. Otherwise, the system located the point pattern by using the HiGrid structure and road intersection directions. Hence, our system did not use HiGrid for maps with known map scales (due to the fact that the system achieves good performance just using the map scale). In general, our approach can also utilize HiGrid for maps with known map scales.

Table 4 shows the performance of GeoPPM with respect to different scenarios. There is only one of our fifty tested maps (i.e., 2%) where the intersection point set is not accurately aligned with the corresponding point pattern on the image. This map is a 1.85 m/pixel resolution TIGER-map with 13 detected intersections (see Fig. 18(a)). As shown in Fig. 18(c), the identified point pattern on the image was shifted one block to the right. We can observe that this misalignment is because the roads on this map are in a grid shape with

Table 4 The performance of GeoPPM

Parameters	Average GeoPPM performance with respect to different map services					Average GeoPPM performance with respect to different regions		Average GeoPPM performance with respect to different resolution maps			
	ESRI map (%)	MapQuest map (%)	Yahoo map (%)	TIGER map (%)	Topographic map (%)	Test Set 1 (El Segundo, CA) (%)	Test Set 2 (St. Louis, MO) (%)	Resolution≤ 2 m/pixel (15 maps) (%)	2 m/pixel< Resolution ≤ 4 m/ pixel (7 maps) (%)	4 m/pixel< Resolution ≤ 7 m/ pixel (13 maps) (%)	Resolution> 7 m/pixel (5 maps) (%)
Precision	96.0	95.2	94.0	84.2[a]	93.9	91.9	93.4[b]	87.4[c]	92.9	96.4	91.6
Recall	80.2	84.8	88.3	75.6[a]	80.94	84.6	77.4[b]	78.2[c]	84.0	88.6	77.1

[a] If we exclude the misaligned TIGER-map, the precision is 93.6% and recall is 84.2%.

[b] If we exclude the misaligned TIGER-map, the precision is 97.2% and recall is 80.6%.

[c] If we exclude the misaligned TIGER-map, the precision is 93.2% and recall is 82.5%.

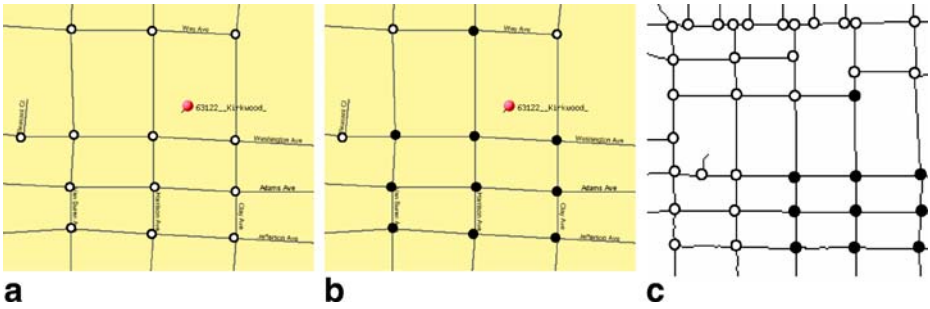


Fig. 18 The map whose detected point pattern does not align with the corresponding point pattern on the imagery. **a** Detected map intersections (white circles). **b** Identified point pattern on the map after applying GeoPPM (point pattern marked as the black circles). **c** Identified point pattern on the imagery after applying GeoPPM (point pattern marked as black circles)

similar block distances and cover a smaller area as compared with other maps. Hence, since there is more than one similar point pattern, the result is ambiguous.

The maps available on TIGER map service and MapQuest are in fixed dimensions. The covered area becomes smaller whenever one zooms in the area of interest. If these small maps contain a unique point pattern across the dataset, GeoPPM can still identify the matched pattern from the maps even with very few points (as in the MapQuest example shown in Fig. 19). However, sometimes, there is no unique pattern (e.g., there are repetitive patterns) on maps covering rather small areas (see Fig. 18(a)). To address this issue, the system can focus on larger maps (i.e., covering larger area) when there is more likely to find a unique pattern of points. We believe that even for the urban areas with repetitive street patterns, a map covering larger areas will produce the distinguishing patterns.

Overall, we make the following observation from Table 4:

- GeoPPM performs well with respect to maps queried from diverse online map services. Additionally, there is no significant difference in the performance over various resolutions of maps. It has the worst performance over TIGER maps because it found one mis-matched point pattern from one of the TIGER maps (i.e., with precision 0% and recall 0%, as shown in Fig. 18).

Fig. 19 The map whose detected point pattern aligns with the corresponding point pattern on the imagery. **a** Detected 16 map intersections (white circles). **b** Identified point pattern after applying GeoPPM (point pattern marked as black circles). **c** Identified point pattern on the imagery after applying GeoPPM (point pattern marked as black circles)

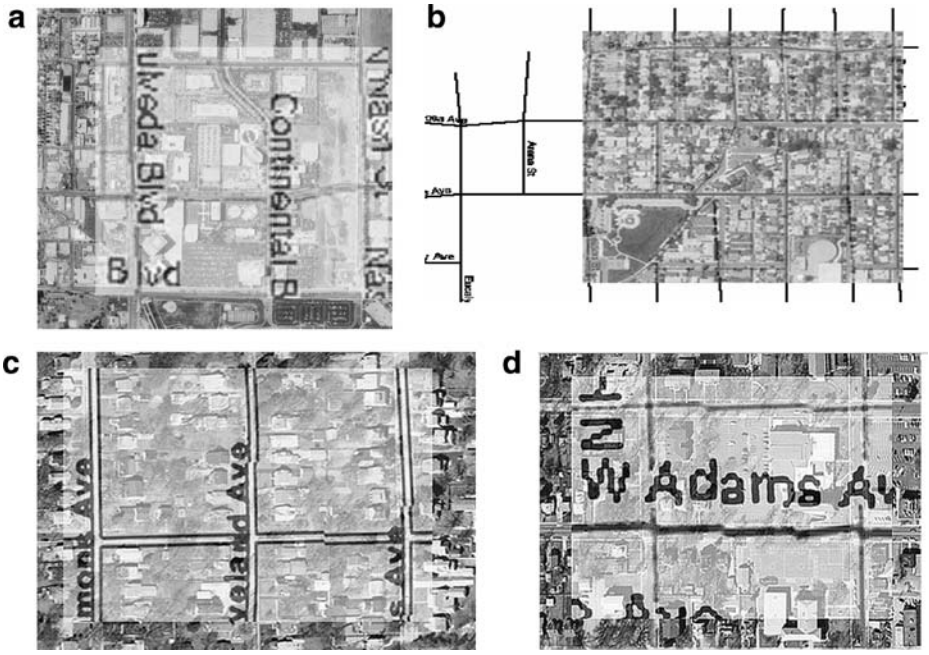


Fig. 20 Examples of map–imagery conflation results. **a** MapQuest map to imagery conflation (semi-transparent map) for El Segundo, CA. **b** TIGER map to imagery conflation (semi-transparent image) for El Segundo, CA. **c** ESRI map to high resolution imagery conflation (semi-transparent map) for St. Louis, MO. **d** MapQuest map to high resolution imagery conflation (semi-transparent map) for St. Louis, MO

5.3.2 Performance of overall map to imagery conflation

After applying GeoPPM, the system generated an accurate control point pair set for each map. Subsequently, our approach used these control points to conflate the maps with imagery. To demonstrate the accuracy of our conflation techniques, some results are shown in Fig. 20. As

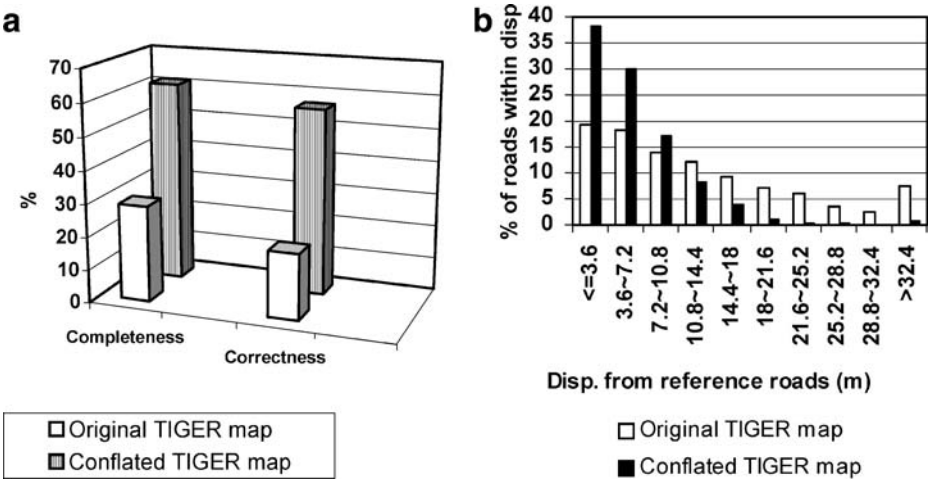


Fig. 21 Map–imagery conflation performance measurement. **a** Completeness and Correctness assessment. **b** Positional accuracy assessment

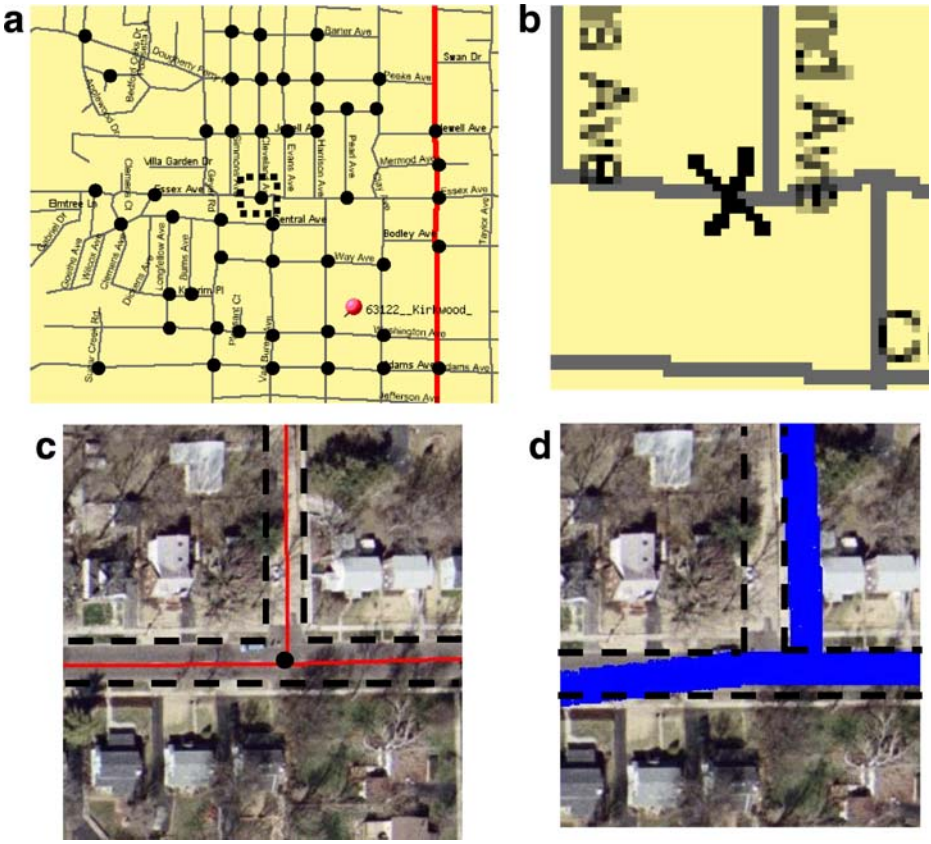


Fig. 22 Explanations of the conflation errors. **a** Detected point pattern on the map (one point is highlighted by dash rectangle). **b** The location of the detected point is marked as "X", after zooming in the highlighted area of (**a**). It is two pixels away from the actual intersection location. **c** The black circle represents the matched imagery point. Its location is very accurate (the solid lines are road vector data and the dash lines are road sides). **d** The two pixels displacement will be amplified due to resizing (the solid regions are map roads and the dash lines are road sides)

shown in these aligned images, the system can annotate spatial objects (e.g., streets) on imagery with the attribution information contained in maps.

We also conducted a quantitative analysis of the conflation results. Towards that end, we randomly selected a set of TIGER maps and imagery from the Test Set 2. These selected maps cover 8.3% of the tested area. After applying GeoPPM against the tested TIGER maps and imagery, GeoPPM accurately obtained aligned control point sets (with 100% precision and 82.7% recall). The reasons why we choose TIGER maps are:

- The geographic coordinates are provided by the data source. Therefore, we can simply combine the TIGER maps with the corresponding imagery based on the provided geographic coordinates. The integration results were then compared with the conflation results utilizing our approach.
- We do not have to specify (i.e., vectorize) the streets on TIGER maps manually for the evaluation purpose, since we can utilize the road vector dataset TIGER/Lines as the vectorized map roads. The roads on the TIGER maps align well with the TIGER/Lines, because they have originally been generated from the TIGER/Lines

vector data set. Hence, we can apply the transformations (formed by control point pairs) to the vector dataset TIGER/Lines. We then evaluate the results by measuring completeness, correctness and positional accuracy to the transformed TIGER/Lines against the reference roads.

As the experimental results shown in Fig. 21, our approach improved the original TIGER map alignment by about 2.5 times for completeness and correctness.

There are three reasons why the completeness and correctness are not that high:

- The errors from original TIGER-maps: For a particular road segment, if the shape of the original TIGER map road is inconsistent with roads in the imagery, our approach may not align them well (although the intersections might be aligned using GeoPPM).
- The errors from resizing: The TIGER-map used in our experiment is 4.17 meters/pixel, while the imagery is 0.3 meter/pixel. After finding the matched point pattern, the system deformed and resized the map to align with the imagery. Due to the large difference in the resolutions of the two datasets, some errors are amplified after resizing. Consider the detected road intersection point shown in Fig. 22(a) and (b). This point highlighted is characterized as an accurately detected map point, because it is two pixels away from the exact position of the intersection (less than the five pixel threshold). Although the imagery intersection is accurately identified (see Fig. 22(c)), the resizing process will make the "2 pixels" displacement become "30 pixels" (as in Fig. 22(d)).
- The errors from the vectorization process: Although the road vector TIGER/Lines align well with the map road, there are still a few pixels difference between them. When utilizing TIGER/Lines as the map roads and resizing to the imagery size, the small errors will also be amplified.

When relaxing the "buffer-width" used to measure completeness/correctness, we can obtain higher completeness/correctness. This assessment is illustrated by the positional accuracy shown in Fig. 21(b). Intuitively, the "positional accuracy" corresponds to the users' demand: *how far is the conflated road network on the map from the centerlines of the real (reference) road network*. We evaluated these displacements between two networks by gradually increasing the buffer-widths constructed around the reference road network. The buffer-width was increased by 3.6 m (i.e., the U.S. standard lane width). As shown in the *X*-axis of Fig. 21(b), the displacement values are grouped every 3.6 m. The *Y*-axis shows the percentage of the conflated map roads lying within the displacement range represented by the *X*-axis. Although our approach did not achieve high completeness/correctness (as stated earlier), it achieved better positional accuracy: 85.2% of the conflated map roads are within 10.8 m of the reference roads compared to 51.7% of the original TIGER map. Furthermore, there are very few road pixels (0.7%) that have more than 32.4 m displacement for conflated roads, compared with 7.5% for the original map. This implies that the conflated map roads are very close to the real roads, although they might not be within the road sides (i.e., road buffer widths).

5.3.3 The execution time

Finally, we evaluated the running time of our conflation algorithm.[17] Since the running time of our techniques is mainly dominated by the point matching routine, we use the running

[17] The experiment platform was a Pentium III 1.2 GHz processor with 512 MB memory running Windows XP (with .NET framework installed).

Table 5 The execution time of GeoPPM

	Using map scale only	Using map scale and road directions
402 imagery points	171 seconds	16 seconds
591 imagery points	317 seconds	26 seconds
800 imagery points	540 seconds	42 seconds
1059 imagery points	934 seconds	70 seconds

a) First scenario: map scale is known (there are 57 map points)

	Brute force algorithm	Using road directions	Using HiGrid and road directions
402 imagery points	5 hours 58 minutes	503 seconds	11 seconds
591 imagery points	N/A*	1049 seconds	17 seconds
800 imagery points	N/A*	2449 seconds	26 seconds
1059 imagery points	N/A*	5298 seconds	38 seconds

b) Second scenario: map scale is unknown (there are 57 map points)

	Using HiGrid and road directions
n = 10	38 seconds
n= 28 (i.e., half of number of map points)	104 seconds
n= 57 (i.e., number of map points)	106 seconds

c) The impact of HiGrid parameter *n*

* Due to the poor performance of brute-force algorithm, we did not collect the running time.

time of GeoPPM routine as the overall execution time (the query time for retrieving online images or maps was not included). In addition, the running time of the GeoPPM algorithm mainly depends on the number of road intersections on the maps, not on the map sizes or map scales. Therefore, we evaluated the time by gradually increasing the number of points on the imagery. In order to compare the time for maps with known map scales and those with unknown map scales, we randomly selected a Yahoo map (with 57 detected points that is close to the average number of intersections of our tested maps) from Test Set 2. We executed the GeoPPM against the Yahoo map using the known map scale and then repeated the same process but assumed that the map scale is unknown. In addition, we recursively partitioned the imagery space into four sub-grids when building the HiGrid structure. We also adjusted the parameter n in Eq. 1 to examine the performance due to its different values. This parameter implies the average number of points in the lowest level determining the depth of HiGrid. Because the number of points on each tested map is rather small compared with the entire set of points in the imagery, we only considered the partitioning of the image space but not the map space. In general, our approach can be used to partition both spaces.

We conflated the Yahoo map with images of varying area sizes with different number of image points. The execution time is shown in Table 5. There are some immediate observations from this table:

- Using map scale information, GeoPPM can significantly improve the execution time.
- For the maps with known scale, the performance when using road directions is significantly better than just using the map scale information.

- Although our approach did not utilize the HiGrid structure for the map with known map scale, it shows similar performance with respect to the one whose map scale is unknown and utilizes HiGrid. Hence, exploiting the map scale is an effective way to match point patterns.
- GeoPPM utilizing HiGrid outperforms the algorithm that just utilizes road directions.
- Although the road direction information significantly improves the brute-force algorithm, it still may need to examine a large number of potential matching point pairs. This results in long running time for datasets with large number of points.
- Using small HiGrid parameter n (i.e., HiGrid with large depth), our approach can efficiently locate the matched point pattern without losing accuracy. This implies that the points are scattered in local areas. In addition, as shown in Table 5(c), there is no significant performance difference for the value 28 (half of the number of map points) and 57 (number of map points), because they result in the same HiGrid depth (according to Eq. 1).

6 Related work

Conflation was first proposed and implemented in 1988 by Saalfeld [25]. The initial focus of conflation was to eliminate the spatial inconsistency between two overlapping vector maps in order to improve the spatial accuracy of vector maps. Once the spatial discrepancy is eliminated, it is possible and easier to transfer attributes among datasets to achieve geospatial data fusion. Several important application domains that can benefit from such data fusion are crisis management, city traffic planning, and military intelligence applications.

In general, based on the types of geospatial datasets dealt with, the conflation technologies can be categorized into following three groups:

- Vector to vector data conflation: For example, the integration of two road networks of different accuracy levels [14], [25], [26], [30], [32].
- Vector to raster data conflation: For example, the integration of road network and imagery [2], [11], [16], [21] or road network and raster maps.
- Raster to raster data conflation: For example, the integration of two images with different resolutions or the integration of raster maps and imagery [15], [27], [29].

In particular, since map-imagery conflation is a sort of raster to raster data conflation, we review related work on raster to raster data integration in details as follows.

Many commercial GIS products, such as Able R2V and Intergraph I/RASC provide the functionality of conflating imagery and maps using different types of transformation methods. However, these products do not provide automatic conflation, so users need to manually pick control points for conflation.

We are not aware of any study addressing the automatic conflation of raster maps and orthoimagery, while there are many studies that focus on imagery to imagery registration. In [27], Sato et al. describe how an edge detection process can be used to determine a set of features that can be used to conflate two image data sets. However, their work requires that the coordinates of both image data sets be known in advance. Our work does not assume that coordinates for the maps are known in advance, although we do assume that we know the general region. Dare and Dowman [15] proposed a feature-based registration technique (based on multi-feature extraction and matching techniques) to integrate two images.

However, their approach requires users to manually select some initial control points to align two images at the first stage. Seedahmed and Martucci [29] proposed an approach, named GIPSC (Geometrically Invariant Parameter Space Clustering), to extract features from imagery by Moravec feature detector and obtain transformation parameters by investigating the strongest clusters in the parameter space. Their approach requires significant CPU time, due to the examination of all potential matching point pairs to find registration parameters. Furthermore, their approach assumes that there exists a global transformation to conflate two images, while we utilized several local transformations to correlate two datasets.

Finally, our GeoPPM algorithm is related to some existing algorithms utilized to solve the correspondence among datasets. One example of such algorithms is RANSAC [17] that randomly selects potential matching points to compute the appropriate transformations. The GeoPPM algorithm significantly differs from RANSAC in terms of our approach to locate potential matching points. More precisely, GeoPPM applies an efficient, systematic and hierarchical approach and exploits prominent spatial properties to search for matched point patterns in local regions. This in turn improves both the accuracy and efficiency, because this exploited knowledge coincides with the natural characteristics of the datasets (i.e., maps and imagery).

7 Conclusion and future work

The main contribution of this paper is the design and implementation of a novel data fusion approach to automatically conflate street maps with orthoimagery. The other contribution is that our technology can process maps that have not been georeferenced and automatically determine their geocoordinates.

We plan to extend our approach in several ways. First, we plan to enhance GeoPPM by first examining the points that have the minimum number of matching candidates. Second, we intend to use OCR-related techniques to extract textual information from the maps. This pre-extracted textual information (such as road names) can be used to label the detected intersections. Therefore, we can even further prune the search space of possible point pattern matching by using these labeled intersections. Third, we would like to apply our technique to other types of maps besides just street maps. This includes a wide variety of maps that are available from various government agencies, such as property survey maps and maps of oil and natural gas fields. Finally, an interesting direction with respect to integrating maps is to be able to take arbitrary maps with unknown map scale and/or geocoordinates and determine their map scale and/or location anywhere within a city, state, country, or even the world. Road vector data covering most of the world is available, so the real challenge is enhancing the HiGrid technique for the point pattern matching to make such a search tractable.

Acknowledgements This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0238560 (PECASE), and IIS-0324955 (ITR), and in part by the Air Force Office of Scientific Research under grant numbers FA9550-04-1-0105, FA9550-07-1-0416 and FA9550-06-C-0120, and in part by the Department of Homeland Security under ONR grant number N00014-07-1-0149.

The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

References

1. M.-F. Aculair-Fortier, D. Ziou, C. Armenakis and S. Wang. "Survey of work on road extraction in aerial and satellite images". Technical Report. Universite de Sherbrooke, (2000).
2. P. Agouris, A. Stefanidis and S. Gyftakis. "Differential Snakes for Change Detection in Road Segments". *Photogrammetric Engineering & Remote Sensing*, Vol. 67(12):1391–1399, December, 2001.
3. T. Barclay, J. Gray and D. Stuz. "Microsoft TerraServer: a spatial data warehouse". In the Proceedings of ACM SIGMOD 2000, 307–318 2000.
4. M.D. Berg, M.V. Kreveld, M. Overmars and O. Schwarzkopf. "Computational geometry: algorithms and applications", Springer-Verlag 1997.
5. R. Cao and C.L. Tan. "Text/graphics separation in maps". In the Proceedings of the 4th International Workshop on Graphics Recognition Algorithms and Applications, Ontario, Canada, pp. 167–177, 2001 September 7–8.
6. D.E. Cardoze and L.J. Schulman. "Pattern matching for spatial point sets". *In the Proceedings of IEEE Symposium on Foundations of Computer Science*, 156–165 1998.
7. C.-C. Chen. "Automatically and accurately conflating road vector data, street maps and orthoimagery". Ph. D. Dissertation. Computer Science Department. University of Southern California. Los Angeles, CA, 2005.
8. C.-C. Chen, C.A. Knoblock and C. Shahabi. "Automatically conflating road vector data with orthoimagery". *Geoinformatica*, Vol. 10(4):495–530, 2006 December.
9. C.-C. Chen, C.A. Knoblock, C. Shahabi, Y.-Y. Chiang and S. Thakkar. "Automatically and accurately conflating orthoimagery and street maps". In the Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS'04), ACM Press, Washington, D.C, pp. 47–56, 2004 November 12–13.
10. C.-C. Chen, C. Shahabi, C.A. Knoblock and M. Kolahdouzan (2006a). "Automatically and efficiently matching road networks with spatial attributes in unknown geometry systems". In the Proceedings of the Third Workshop on Spatio-Temporal Database Management (co-located with VLDB2006), Seoul, Korea, pp. 1–8, September 2006.
11. C.-C. Chen, S. Thakkar, C.A. Knoblok and C. Shahabi. "Automatically annotating and integrating spatial datasets". In the Proceedings of the International Symposium on Spatial and Temporal Databases, LNCS 2750,Springer-Verlag, Santorini Island, Greece, pp. 469–488, July 24–27, 2003.
12. L.P. Chew, M.T. Goodrich, D.P. Huttenlocher, K. Kedem, J.M. Kleinberg and D. Kravets. "Geometric pattern matching under Euclidean motion". In the Proceedings of the Fifth Canadian Conference on Computational Geometry, pp. 151–156, 1993.
13. Y.-Y. Chiang, C.A. Knoblock and C.-C. Chen. "Automatic extraction of road intersections from raster maps". In the Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems, Bremen, Germany, pp. 267–276, November 4–5th, 2005.
14. M. Cobb, M.J. Chung, V. Miller, H.I. Foley, F.E. Petry and K.B. Shaw. "A rule-based approach for the conflation of attributed vector data". *GeoInformatica*, Vol. 2(1):7–35, 1998.
15. P. Dare and I. Dowman. "A new approach to automatic feature based registration of SAR and SPOT images". *International Archives of Photogrammetry and Remote Sensing*, Vol. 33(B2):125–130, 2000.
16. S. Filin and Y. Doytsher. "A linear conflation approach for the integration of photogrammetric information and GIS data". *International Archives of Photogrammetry and Remote Sensing*, Vol. 33:282–288, 2000.
17. M.A. Fischler and R.C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, Vol. 24 (6):381–395, 1981.
18. M. Flavie, A. Fortier, D. Ziou, C. Armenakis and S. Wang. "Automated updating of road information from aerial images". In the Proceedings of American Society Photogrammetry and Remote Sensing Conference, Amsterdam, Holland, July, 16–23, 2000.
19. M.F. Goodchild and G.J. Hunter. "A simple positional accuracy measure for linear features". *International Journal of Geographic Information Sciences*, Vol. 11(3):299–306, 1997.
20. A. Habib, Uebbing, R., Asmamaw, A. "Automatic extraction of primitives for conflation of raster maps". Technical Report. The Center for Mapping, The Ohio State University, 1999.

21. H. Hild and D. Fritsch. “Integration of vector data and satellite imagery for geocoding”. *International Archives of Photogrammetry and Remote Sensing*, Vol. 32(Part 4):246–251, 1998.
22. J.-R. Hwang, J.-H. Oh and K.-J. Li. “Query transformation method by Delaunay triangulation for multi-source distributed spatial database systems”. In the Proceedings of the 9th ACM Symposium on Advances in Geographic Information Systems, pp. 41–46, November 9–10, 2001.
23. S. Irani and P. Raghavan. “Combinatorial and experimental results for randomized point matching algorithms”. *Computational Geometry*, Vol. 12(1–2):17–31, 1999.
24. M.T. Musavi, M.V. Shirvaikar, E. Ramanathan and A.R. Nekovei. “A vision based method to automate map processing”. *Pattern Recognition*, Vol. 21(4):319–326, 1988.
25. A. Saalfeld. “Conflation: automated map compilation”. *International Journal of Geographic Information Sciences*, Vol. 2(3):217–228, 1988.
26. A. Saalfeld. “Conflation: automated map compilation”. Computer Vision Laboratory, Center for Automation Research, University of Maryland, 1993.
27. T. Sato, Y. Sadahiro and A. Okabe. “A computational procedure for making seamless map sheets”. Technical Report. Center for Spatial Information Sciences, University of Tokyo, 2001.
28. T.J. Sebok, L.E. Roemer and J. Malindzak, G.S. “An algorithm for line intersection identification”. *Pattern Recognition*, Vol. 13(2):159–166, 1981.
29. G. Seedahmed and L. Martucci. “Automated image registration using geometrically invariant parameter space clustering (GIPSC)”. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 34(3A):318–323, 2002.
30. V. Walter and D. Fritsch. “Matching spatial data sets: a statistical approach”. *International Journal of Geographic Information Sciences*, Vol. 5(1):445–473, 1999.
31. M.S. White and P. Griffin. “Piecewise linear rubber-sheet map transformation”. *The American Cartographer*, Vol. 12(2):123–131, 1985.
32. S. Yuan and C. Tao. “Development of conflation components.”*In the Proceedings of Geoinformatics*, pp. 19–21, 1999.

Ching-Chien Chen is the Director of Research and Development at Geosemble Technologies. He received his Ph.D. degree in Computer Science from the University of Southern California for a dissertation that presented novel approaches to automatically align road vector data, street maps and orthoimagery. His research interests are on the fusion of geographical data, such as imagery, vector data and raster maps with open source data. His current research activities include the automatic conflation of geospatial data, automatic processing of raster maps and design of GML-enabled and GIS-related web services. Dr. Chen has a number of publications on the topic of automatic conflation of geospatial data sources.

Craig Knoblock is a Senior Project Leader at the Information Sciences Institute and a Research Professor in Computer Science at the University of Southern California (USC). He is also the Chief Scientist for Geosemble Technologies, which is a USC spinoff company that is commercializing work on geospatial integration. He received his Ph.D. in Computer Science from Carnegie Mellon. His current research interests include information integration, automated planning, machine learning, and constraint reasoning and the application of these techniques to geospatial data integration. He is a Fellow of the American Association of Artificial Intelligence.

Cyrus Shahabi is currently an Associate Professor and the Director of the Information Laboratory (InfoLAB) at the Computer Science Department and also a Research Area Director at the NSF's Integrated Media Systems Center (IMSC) at the University of Southern California. He received his B.S. degree in Computer Engineering from Sharif University of Technology in 1989 and his M.S. and Ph.D. degree in Computer Science from the University of Southern California in 1993 and 1996, respectively. He has two books and more than hundred articles, book chapters, and conference papers in the areas of databases, GIS and multimedia. Dr. Shahabi's current research interests include Geospatial and Multidimensional Data Analysis, Peer-to-Peer Systems and Streaming Architectures. He is currently an associate editor of the IEEE Transactions on Parallel and Distributed Systems (TPDS) and on the editorial board of ACM Computers in Entertainment magazine. He is also in the steering committee of IEEE NetDB and ACM GIS. He serves on many conference program committees such as ACM SIGKDD 2006, IEEE ICDE 2006, ACM CIKM 2005, SSTD 2005 and ACM SIGMOD 2004. Dr. Shahabi is the recipient of the 2002 National Science Foundation CAREER Award and 2003 Presidential Early Career Awards for Scientists and Engineers (PECASE). In 2001, he also received an award from the Okawa Foundations.