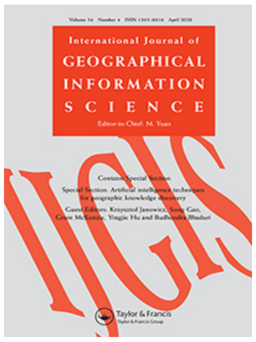

International Journal of Geographical Information Science

ISSN: 1365-8816 (Print) 1362-3087 (Online) Journal homepage: https://www.tandfonline.com/loi/tgis20

Automatic alignment of contemporary vector data and georeferenced historical maps using reinforcement learning

Weiwei Duan, Yao-Yi Chiang, Stefan Leyk, Johannes H. Uhl & Craig A. Knoblock

To cite this article: Weiwei Duan, Yao-Yi Chiang, Stefan Leyk, Johannes H. Uhl & Craig A. Knoblock (2020) Automatic alignment of contemporary vector data and georeferenced historical maps using reinforcement learning, International Journal of Geographical Information Science, 34:4, 824-849, DOI: 10.1080/13658816.2019.1698742

To link to this article: https://doi.org/10.1080/13658816.2019.1698742

Published online: 09 Dec 2019.

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
2020, VOL. 34, NO. 4, 824–849
https://doi.org/10.1080/13658816.2019.1698742

RESEARCH ARTICLE

Automatic alignment of contemporary vector data and georeferenced historical maps using reinforcement learning

Weiwei Duan[a], Yao-Yi Chiang[b], Stefan Leyk[c], Johannes H. Uhl[c] and Craig A. Knoblock[d]

[a]Department of Computer Science, University of Southern California, Los Angeles, USA; [b]Spatial Sciences Institute, University of Southern California, Los Angeles, USA; [c]Department of Geography, University of Colorado Boulder, Boulder, CO, USA; [d]Information Sciences Institute, University of Southern California, Los Angeles, USA

ABSTRACT
With large amounts of digital map archives becoming available, automatically extracting information from scanned historical maps is needed for many domains that require long-term historical geographic data. Convolutional Neural Networks (CNN) are powerful techniques that can be used for extracting locations of geographic features from scanned maps if sufficient representative training data are available. Existing spatial data can provide the approximate locations of corresponding geographic features in historical maps and thus be useful to annotate training data automatically. However, the feature representations, publication date, production scales, and spatial reference systems of contemporary vector data are typically very different from those of historical maps. Hence, such auxiliary data cannot be directly used for annotation of the precise locations of the features of interest in the scanned historical maps. This research introduces an automatic vector-to-raster alignment algorithm based on reinforcement learning to annotate precise locations of geographic features on scanned maps. This paper models the alignment problem using the reinforcement learning framework, which enables informed, efficient searches for matching features without pre-processing steps, such as extracting specific feature signatures (e.g. road intersections). The experimental results show that our algorithm can be applied to various features (roads, water lines, and railroads) and achieve high accuracy.

ARTICLE HISTORY
Received 15 May 2018
Accepted 25 November 2019

KEYWORDS
Vector-to-raster alignment; reinforcement learning; USGS historical topographic maps; digital map processing; digital humanities

1. Introduction

Historical maps store valuable information documenting human activities and natural features on Earth over long time periods (Uhl *et al.* 2018b). With increasing access to inexpensive and efficient digitalization technology, many organizations have scanned tens of thousands of historical maps and stored them in digital map archives. While these efforts ensure the preservation of important historical documents, using the map contents as geographic information layers in an analytic environment requires advanced technologies for the automatic detection and vectorization of geographic features from

CONTACT Weiwei Duan weiweidu@usc.edu

large numbers of map images (Chiang *et al*. 2014, Chiang and Knoblock 2015, Ostafin *et al*. 2017). Convolutional Neural Networks (Krizhevsky *et al*. 2012, Razavian *et al*. 2014, Simonyan and Zisserman 2015, Chen *et al*. 2018) have shown impressive performance in object segmentation (i.e. detecting precise locations of individual objects in images) in public image recognition datasets (such as PASCAL-Context[1]) and more recently, in extracting geographic features from historical scanned maps (Duan and Chiang 2017b, Uhl *et al*. 2018a). However, one of the major challenges in using CNN for map processing is how to efficiently generate a sufficient amount of representative training data. Manually annotating training samples can be accurate but highly time-consuming.

Recent studies have developed a segmentation model for railroad features using more than 10,000 training samples (Duan *et al*. 2017a) and identified a need to generate a much larger amount of representative training data automatically. One possible solution to facilitate the automatic collection of training samples is the use of data representing the geographic feature of interest from other sources to annotate the corresponding locations in historical maps. In this approach, the system can label training samples automatically using these external data, assuming they are aligned well with the content in the map. For example, Figure 1 demonstrates how to use external data to generate annotations for training samples. In Figure 1(a), the violet lines are the external railroad vector features, and the black lines with crosses in the underlying historical map image are the railroad feature representations. Figure 1(b) is a training sample, within which the content

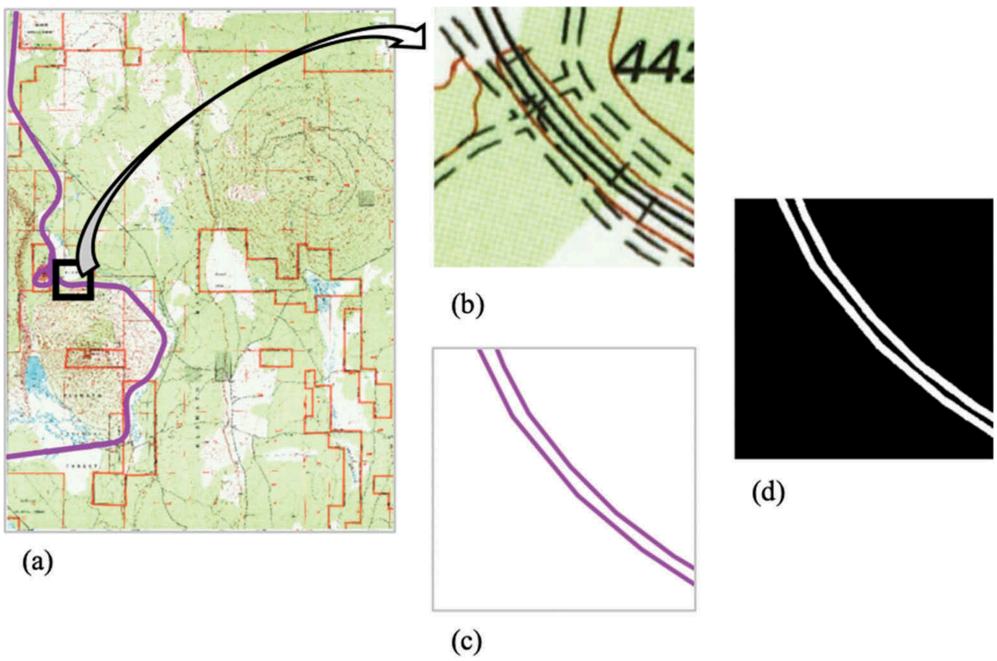

Figure 1. An example demonstrating how to automatically generate railroad training samples for detecting railroads using auxiliary vector data. (a) Historical map and overlaid contemporary railroad data (the violet line) assumed to be aligned with railroads in a historical map, (b) map sample cropped in a focal window along the railroad line, showing the railroad symbol in the map as a black line with crosses, (d) is the labels for all pixels in (b). The pixel labeled as white in (d) if it overlaps with the vector data (shown in (c)), otherwise black.

of the map is extracted by the vector data shown in Figure 1(c). The extracted contents are used to create labels for pixels on the map. Figure 1(d) is a label for the training sample (Figure 1(b)). White color represents railroad pixels, while black means non-railroad pixels. However, often contemporary vector data (e.g. from the US Census Bureau) can only provide the approximate locations of geographic features in scanned historical maps (Figure 2) due to misalignment.

There are three main causes of this misalignment. First, distortion can be introduced by projection inconsistency, georeferencing, and scanning (Uhl *et al.* 2017, 2018b). The projection and the geodetic datum may be different between the two datasets. These differences can result in feature misalignment when converting from one projection to another (Figure 2(a)). Distortions can further be elevated by deformations of the paper map, scanner mis-calibration, and inaccuracies in the georeferencing process (Uhl *et al.* 2018b). Second, datasets can adopt different geometric representations for the same geographic features. For example, in Figure 2(b), the streams are represented as blue curved lines on the map but straight lines in the vector data, due to different degrees of cartographic generalization (shown in red). Third, geographic features change over time. For example, Figure 2(c) shows an example in which the contemporary vector data (the red line) represent the new road that did not exist in the historical map. Or, a railroad could exist in the historical map but not in the contemporary data if it has been removed. To overcome the positional offset problems between maps and auxiliary data, in this paper we introduce a vector-to-raster alignment algorithm, which aligns georeferenced

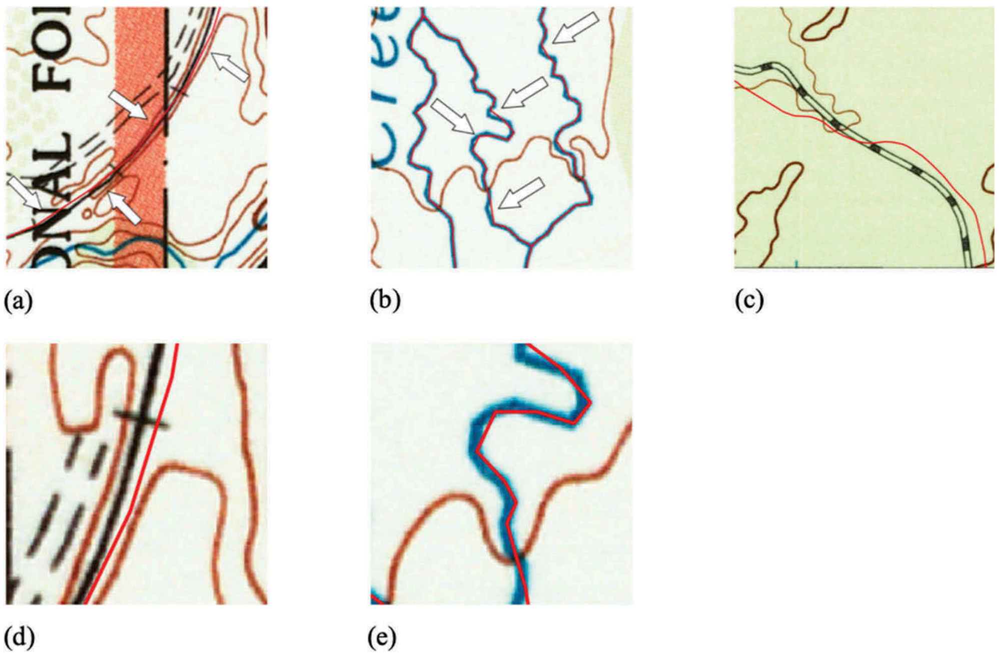

Figure 2. Examples show the misalignment between contemporary vector data (red lines) and content in historical maps (pointed at by white arrows). The misalignment could be caused by (a) different spatial reference systems, (b) different feature representation, or (c) inconsistent content due to different publication dates e.g. new roads. (d) and (e) are a zoom-in figures of (a) and (b) respectively.

contemporary vector data to scanned historical maps, thus supporting automatic training data collection for feature extraction from maps.

The proposed vector-to-raster alignment algorithm, based on a reinforcement learning framework, aims at matching georeferenced vector data with their corresponding representations in georeferenced map images. Previous studies (Chen *et al*. 2006, Song *et al*. 2013, Ruiz-Lendínez *et al*. 2017) have focused on the identification of 'feature signatures' to represent the geographic features and used these signatures (e.g. intersections of roads) as the key elements for the matching process. However, the use of feature signatures constrains the applicability of the methods to few geographic features. Other work (e.g. Tong *et al*. 2009) tested matching representations of different types of geographic features, requiring human intervention to generate the feature signatures. Human intervention makes the methods impractical if the goal is to scale the process to thousands of maps in a typical map archive. The presented vector-to-raster alignment algorithm is designed to handle various geographic features without human interventions. Developing such an algorithm raises the question on how to automatically and efficiently match geographic features of various representations found in heterogeneous data sources (e.g. matching railroads in the historical scanned map and contemporary vector data) and what matching criteria could be defined to guide this process. The reinforcement learning framework provides a possible solution to these questions. In this framework, an agent finds an efficient solution to a problem, guided by rewards. For example, to enable a robot to find the shortest path from the entrance to the exit in a maze, the robot receives a reward for each step taken. The rewards will guide the robot to find the shortest path, efficiently. After multiple trials in discovering possible paths to the exit, the robot will identify and memorize the shortest path to exit the maze. For the vector-to-raster alignment problem, a reward is defined based on the matching quality between the vector feature and the geographic feature of interest in the map image. The alignment process guided by the rewards is more efficient than a brute-force search for matching and does not require human intervention or specific feature signatures (e.g. road intersections). We will explain the reinforcement learning framework in detail in Section 3 and describe its applicability to the vector-to-raster alignment problem in Section 4. Section 5 will show the alignment results of contemporary vector data with their respective representations in georeferenced historical maps for railroads, hydrographic features (streams), and roads.

2. Related work

In this paper, we present an algorithm that automatically aligns vector data to geographic features in georeferenced historical scanned maps. Some prior studies on such related tasks focus on the conflation problems (Chen *et al*. 2006), registration and co-registration problems (Le Moigne *et al*. 2002), and other specific challenges (Wu *et al*. 2007, Wu 2011). Conflation aims to generate a new dataset by integrating multiple input datasets representing the same region, while registration attempts to geocode an image to a specified spatial reference system. In contrast, the goal of alignment in a more geospatial sense is to make multiple geographic layers align correctly. Because of the commonalities between these approaches, we briefly summarize the existing work for conflation, registration, and alignment next.

Some existing methods (Chen *et al*. 2006, 2014, Wu *et al*. 2007, Wu 2011, Song *et al*. 2013) specifically work on roads. They used the intersection as the specific feature signature to integrate road vector data with high-resolution images. The common limitation of their methods is that they can only apply to roads or other geographic features containing intersections. Chen *et al*. (2006) proposed an algorithm that can automatically conflate road vector data with ortho-imagery. Their method first detects matching intersection point pairs from the two datasets and then generates a local transformation matrix for conflation based on the matched pairs. Their method can only apply to roads because they use intersections, the specific characteristic of roads, to align road vector data with imagery. Some other methods (Song *et al*. 2013, Chen *et al*. 2014) used different approaches to detect intersections of roads. For example, Chen *et al*. (2014) used a binary road mask to extract all road-like pixels and then used a junction template to classify road vector points as junction or non-junction. Wu *et al*. (2007) proposed an algorithm dealing with the alignment of road vector data and aerial imagery. Their method used a least-squares optimization to find the best translation for the road vector data to align with the roads on imagery. However, their algorithm also used the road intersection as the specific signature to detect roads on imagery. Compared to their methods, our approach can apply to various types of geographic features (e.g. railroads, water lines, roads).

Although some alignment approaches (Tong *et al*. 2009) can apply to various geographic features, they involve manual steps. Tong *et al*. (2009) built a probability-based multi-measure feature matching method for map conflation. However, their approach requires a manual step to extract representations for geographic features. In contrast, our methods do not require human intervention.

Researchers also use agent-based models to solve problems in geographic sciences. Agent-based modeling and reinforcement learning share similarities. For example, Touya *et al*. (2018) used multi-agent systems for cartographic generalization. The goal of cartographic generalization is to derive a smaller-scale map from a large-scale map by transforming the geographic features on the large-scale map to adjust them into the small scale map representations without feature conflicts such as buildings that overlap with roads. In a multi-agent system, for example, one agent represents buildings, and the other represents roads. The agents take actions to resolve the conflict. The knowledge base used by the agents helps decide what action the agents will take. After taking action, the agents estimate if they are closer to the aim of resolving the conflict. Since the system has multiple agents, actions taken by an agent can be influenced by other agents. Each agent in multi-agent systems can be implemented by reinforcement learning when modeling agent behavior and learning based on actions of the agent. In contrast to such complex multi-agent systems, for our alignment problem, there is only one agent, the external vector data. Vector data take actions to align with the geographic feature of interest on the map using reinforcement learning as explained in more detail below.

3. Introduction of reinforcement learning

Reinforcement learning (Kaelbling *et al*. 1996, Kober *et al*. 2013, Sutton and Barto 2018) is a machine learning process in which an agent interacts with the environment by taking actions to earn the maximum cumulative rewards. A reinforcement learning model uses the reward as the signal, which is different from the supervising function in supervised learning, to indicate how well the agent is doing after each step. Here, we briefly introduce the terminology used in the reinforcement learning framework and use the maze problem as an example to show that the reinforcement learning can efficiently find solutions.

3.1. Agent

An agent takes actions within the environment. The learning algorithm controls the agent. For example, the robot is the agent in the maze example.

3.2. Environment

The environment is where the agent takes actions. In our example, the maze is the environment. The environment defines the current state and, based on an action of the agent as the input, allows computing the reward and returns the next state.

3.3. State

A state is a situation for the agent in the environment. For the maze problem, a location of the robot in the maze is a state.

3.4. Episode

An episode is a set of actions from the starting state to the terminal state. For the maze problem, the starting state is the robot located at the entrance, and the terminal state is the robot located at the exit.

3.5. Action

An action is a movement that the agent can take at a given state in the environment. For the maze problem, the robot can move up, down, left or right, or stay at the same location.

3.6. Reward

A reward is a value that measures the contribution of an action taken by the agent at a given state to accomplish the goal.

An advantage of reinforcement learning is that it can efficiently find the solution to the problem. Solving the maze problem is a typical learning process that can greatly benefit from reinforcement learning. In each episode, the robot moves from the entrance (the start state) to the exit (the terminal state) receiving the reward in each state. After several episodes, the robot finds the shortest trajectory from the entrance to exit according to the cumulated rewards. Osmanković and Konjicija (2011) prove that the learning algorithm (the Sarsa algorithm created by Sutton (1996)) reduces the search time for finding the optimal trajectories in mazes of different sizes, compared to brute-force methods. Table 1 compares the number of iterations using the reinforcement learning method and the brute force search.

Table 1. The average number of iterations based on the Sarsa algorithm compared with brute-force searches.

Maze Size	Iterations for reinforcement learning	Iterations for brute-force search in the worst case
5*5	2587.4	$2.2*10^{16}$
7*7	3972.5	$6.3*10^{25}$
9*9	5822.7	$1.2*10^{43}$
11*11	10,755.4	$1.4*10^{67}$

4. Vector-to-raster alignment algorithm based on reinforcement learning

This section first introduces how to model the vector-to-raster alignment problem using the reinforcement learning framework, then defines the reward function, and presents the proposed alignment algorithm.

4.1. Modeling the alignment problem within the reinforcement learning framework

4.1.1. Agent

The agent is a subset of the entire input vector data. For example, the entire railroad vector data to be aligned with a map can be divided into five subsets. Each subset is an agent covering a neighborhood in the map

4.1.2. Environment

The historical scanned map is the environment within which the vector data moves.

4.1.3. State

The position of the vector data on the scanned map is a state. The starting state for our alignment problem is the position of the contemporary vector data on the map (potentially misaligned). The terminal state is the position of vector data aligned to the geographic feature of interest in the map.

4.1.4. Episode

One episode is the process of aligning a subset of the entire input vector data (i.e. an agent) to the geographic feature of interest on the map.

4.1.5. Action

Action is the movements of all vector points in the vector data. We define nine possible movements for each vector point: staying at the current position and moving towards any of the cardinal or intercardinal directions (north, south, west, east, northwest, northeast, southwest, and southeast). The eight directions are assumed to be sufficient to guide the vector data to find the corresponding geographic feature on the map but could be increased as needed. However, more than eight directions might not help improve the alignment performance but take more time to process. Section 4.3 will discuss the number of movement directions as a factor that influences the efficiency of our algorithm.

Figure 3 illustrates examples of states and their possible actions. The green line with red points is the railroad vector data. The black line with crosses is a segment of the

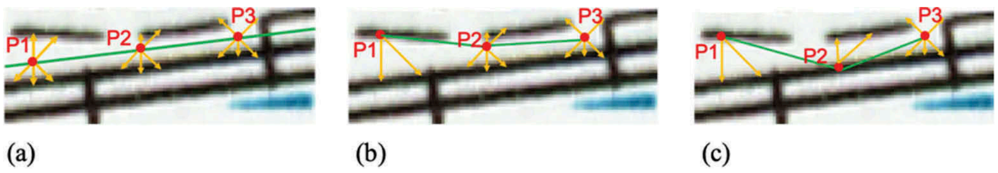

Figure 3. Examples of states and actions. The green line is the railroad vector data, red points are vector points, and yellow arrows are possible movements. The black lines with crosses below the green line are railroad lines. (a) is the starting state. In (b), the action is P1 moves towards north, P2 and P3 stay at the original position. The action in (c) is P1 moves towards north, P2 moves towards south, and P3 stays at the original position.

railroad line on the scanned map. The goal is to align the vector line to the railroad on the map. The yellow arrows at each point denote the available movements that each vector point can take. Besides the movements denoted by the yellow arrows, each point could stay at the original position. The action is the movements of all points. Figure 3(b,c) show examples of one action. For example, the action in Figure 3(b) is that P_1 moves North, P_2 and P_3 stay in the original position.

Here we briefly explain the method for finding the possible movements for each vector point for the vector-to-raster alignment.[2] First, the color range of the geographic feature of interest will be detected. Second, the locations of the geographic features (the foreground) in the map images will be extracted around each vector point. Third, possible movements will be identified for each vector point.

The dominant foreground color of the pixels close to the vector data should represent the corresponding feature symbol (if present) since the vector data are assumed to be close to the corresponding geographic feature in the map. Hence, the method uses ‘growing’ buffers from the vector data to identify the dominant color of nearby pixels as the color of the corresponding geographic feature. The dominant color in the first few buffers is presumed to be the color of the corresponding geographic feature. When the buffer size increases, the dominant color changes to the background color or the color of another nearby geographic feature. Figure 4 illustrates the process of detecting the railroad color. The second step takes the color range of the geographic feature of interest and a sub-image as inputs. Each sub-image is generated by using a vector point as the center to crop the map image within a defined window. Thus, a sub-image represents the neighborhood of a vector point location on a map, which is expected to include the target geographic feature. Figure 5(a) is an example of a sub-image. Our method applies the GrabCut algorithm (Rother *et al.* 2004) on sub-images to extract geographic features around vector points (Figure 5(b)). In the third step, if a geographic feature could be extracted in the second step in a certain cardinal or intercardinal direction of the vector point located at the center of the sub-image, our method defines a movement for that vector point towards the closest graphic feature along the same direction. If no geographic feature has been detected by step two along any given direction, movement of the vector point in that direction will not be considered. Figure 5(c) shows an example of available movements of a vector point. Each arrow represents a possible movement. The vector point cannot find any foreground pixel in the west and northwest directions, eliminating those directions as possible movements. If a vector point does not have any movements except staying at the original position on the map, our algorithm assumes it is a new feature that does not exist in the map and removes it.

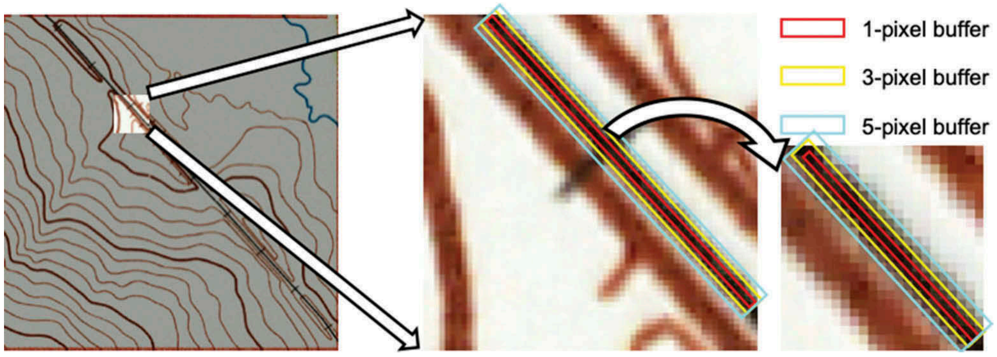

Figure 4. The illustration of detecting the color of railroads using growing buffers. When the buffer size increases, the dominant color changes. The red, yellow, and blue polygons show the overlapping areas of 1-, 3-, and 5-pixel buffers and the map, respectively. When the buffer is 1 pixel and 3 pixels, the dominant color is the color of railroads. When the buffer extends to 5 pixels, there are more background pixels. Since the railroad vector data are around the railroads in the map, the dominant color in the first few buffers (1-pixel and 3-pixel buffer) are assumed to be the railroad color.

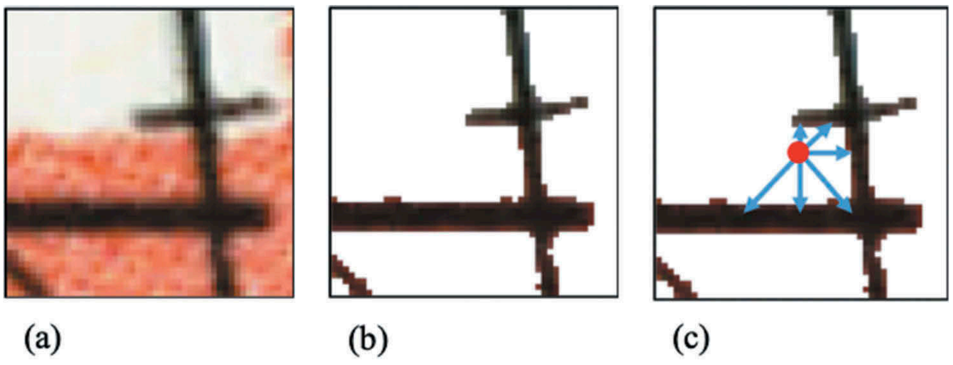

Figure 5. An example of how our method finds possible movement directions for a vector point. (a) shows a sub-image using a railroad vector point as the center. (b) is the foreground of (a) after applying Grabcut. In (c), the blue arrows represent possible movements for the vector point (the red dot in the center).

4.2. The reward function

The reward function in our alignment algorithm guides the agent (the vector data) so that the process of aligning the vector features to the feature representation in the map is efficient. The design of the reward function is based on the color principle in cartography. Color is a distinguishable property for geographic features and remains consistent over map editions. For example, the United States Geological Survey (USGS) topographic map archive uses blue for hydrography, black for roads, and brown for contour lines. Hence, the intuition of the reward design is that the colors of the pixels underlying the vector line on the map should be consistent and describe the color of the corresponding geographic feature if the vector data align perfectly with the geographic feature in the map.[3] Alternatively, the most straightforward way to define the reward is to check if all pixels on the map overlapping with vector data are the color of the geographic feature of interest. However, the movements found by the method introduced in the previous subsection cannot guarantee to align vector data to the centerline of the geographic feature. Some vector segments may align with the edge of the geographic feature. In such cases, some background pixels surrounding the geographic feature of interest on the map overlap with the re-positioned vector data. For example, in Figure 6, some vector segments (the green line) align with the background on the map in the right enlarged view. Therefore, determining the pixel colors on the map that overlap with the vector data as the reward is not accurate in this case. The reward function assigns negative values when a given number of background pixels align with some vector segments even though the vector data also align with the corresponding geographic feature on the map. Since this edge problem is often minor, the reward function should have the capacity to ignore a few background pixels aligned with the vector data.

Instead of directly using the colors of individual pixels underlying the vector segment to generate the reward, our algorithm groups the pixel colors into clusters using a standard K-mean algorithm to detect the dominant pixel colors overlapping with vector segments. If one of the cluster centroids is out of the color range of the target map feature, it indicates that some large portion of the vector segment is not well aligned with the target map feature. Figure 7 shows the pseudocode of the reward function. If a centroid is not within the color range of the target map feature, the reward for the segment is 0; otherwise, it is 1.

4.3. *The vector-to-raster alignment algorithm*

Figure 8 shows the pseudocode of our alignment algorithm. The algorithm takes georeferenced vector data and scanned topographic historical maps made based on modern cartography as inputs and outputs the aligned vector features. Since the features in two datasets can have significantly different geometric properties (e.g. straight lines vs. curves), our alignment algorithm first densifies (i.e. increases the resolution by increasing the

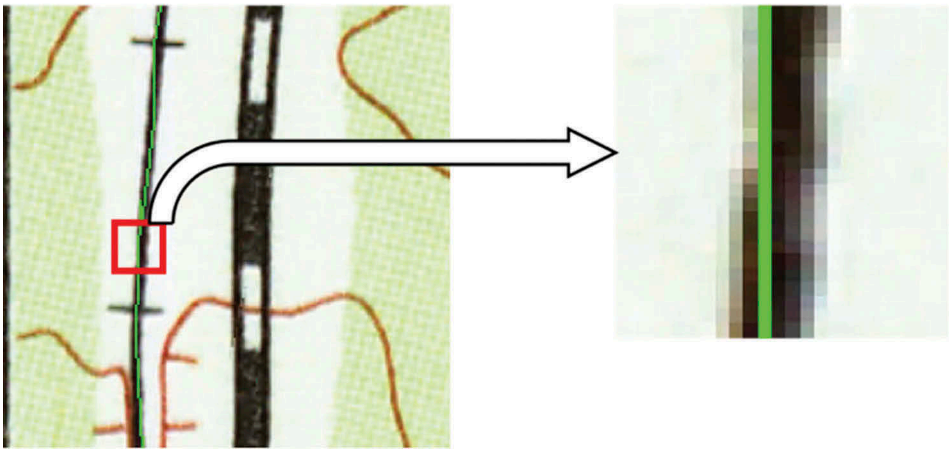

Figure 6. The green line is the aligned railroad vector data. The right figure is the enlarged view of the sub-image within the red box in the left figure. The line segment aligns with the edge of the railroad line on the map, which leads to some background pixels overlapping with the vector data.

```
Reward ($start\_point$, $end\_point$):
    color_list = color($state\_point$, $end\_point$)
    centroids = Kmean($color\_list$, 2)
    if any centroid not in the range of the geographic feature color:
        return 0
    return 1
```

Figure 7. The pseudocode of the reward function.

```
Input: a subset of vector data $V$ containing a set of $points$, the geographic feature $F$,
a scanned map $M$
Output: aligned vector data
$[p_1, p_2, ..., p_n]$ = Interpolation($points$, $interpolate\_distance$)
Randomly choose action $a$ in the starting state
While $V$ do not align with $F$:
    Take action $a$ in $s$, observe the reward $r$, the next state $s'$
    Choose next action $a' = max$[Reward_vector($s'$ after taking $a'$)],
    $s \leftarrow s', a \leftarrow a'$
Align $V$ with $F$ according to $a$

Reward_vector($vector$):
    for each $segment$ $S_i$ in $vector$:
        $R_i$ = Reward($S_i$)
    return $\frac{\sum R_i}{\#Segments}$
```

Figure 8. Our vector-to-raster alignment algorithm.

number of vertices) each segment of the vector polylines so that our algorithm can accommodate different representations of the same geographic feature. For example, there are more curves in water lines than railroads, so our algorithm densifies the water-lines vector data by interpolating vector points every 15 pixels after overlapping the vector data with the map, while interpolating vector points every 30 pixels for railroad vector data. Figure 9 shows an example of densifying vector points in the water-line vector data to accommodate the sinuosity of water lines in the map. After the densification process, the

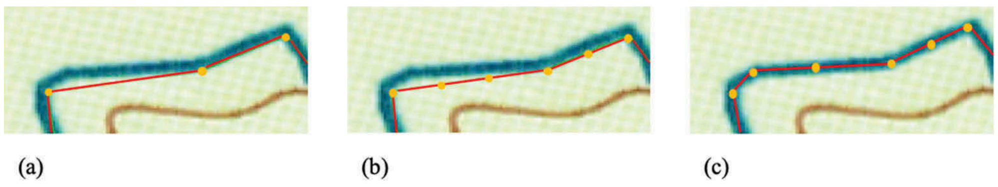

Figure 9. The densification process helps to accommodate the straight line segments representing water lines in vector data to the curved representations (the blue lines) in the scanned map. (a) shows the vector data (the red line) and the waterline (the blue line) on the map. (b) is the vector data after densification. (c) shows the vector data (the red line) aligned to the water line represented as the curve in the map.

vector data take actions on the map guided by the reward function. Our algorithm gives each state (the position of vector polyline on the map) a reward to quantify how close the state to the final alignment state is after taking action. The reward for a vector polyline is the sum of all individual rewards of vector segments divided by the total number of segments in the vector polyline. Our algorithm calculates the reward for each segment in the vector polyline by the reward function introduced in Section 4.2. The reward ranges from zero to one. Our alignment algorithm always chooses the action with the maximal cumulative reward. In contrast, traditional reinforcement learning methods such as the Q-learning approach proposed by Watkins and Dayan (1992), typically have a small probability of choosing a random action. Our goal is to find a solution to align the vector data to the corresponding geographic feature on the map accelerated by the reward function, while the goal for Q-learning is finding the fastest solution. Consequently, our algorithm only requires running one episode instead of multiple episodes to find the fastest alignment solution. Therefore, using the greedy method to choose the actions is more efficient for our problem than the method for Q-learning.

We use an example in Figure 10 to illustrate our alignment algorithm. The two black lines are the geographic features on the map, the green line is the vector data, and the red points are the vector points. The goal is to align the green vector line with the black bottom line. Given that the target representation is south of the green line and without loss of generality, the points here only have five types of possible movements: staying at the current position, represented as origin, or moving towards four possible cardinal directions (north, south, west, and east). Actions are the set of movements of all points represented by a list. The green line aligns with the black line after three iterations in Figure 10. In each iteration, our algorithm tries to move one vector point to maximize the reward and records all movements of vector points as the action for the state. Using reinforcement learning to search an alignment solution is much more efficient than using the brute-force search. For example, if we use brute-force search for alignment in Figure 10, we need to run a maximum of possible 125 iterations (three points, each has five possible actions, so the total possible actions are 5^3) to align the vector line with the geographic feature in the map. In contrast, our vector-to-raster alignment algorithm can solve this problem in three iterations, guided by rewards.

In addition to the reward function there are three other major factors that have an impact on the efficiency of our algorithm. To discuss the efficiency of the algorithm, we assume that our algorithm applies to the same geographic feature in the same region,

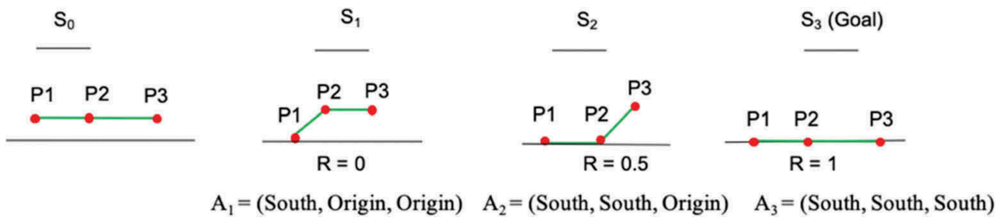
S_0
S_1
S_2
S_3 (Goal)
P1
P2
P3
R = 0
R = 0.5
R = 1
A_1 = (South, Origin, Origin)
A_2 = (South, South, Origin)
A_3 = (South, South, South)

Figure 10. An example of the alignment process. In the action set (A1, A2, A3), each element represents the movement of vector point sharing the same index. For example, (South, Origin, Origin) in A1 represents that P1 moves towards South, P2 and P3 stay at the original position. The starting state does not have a reward, because there is no action in the starting state.

which may have different representations on different maps. The first factor is the number of vector points (vertices) during the densification process. Our algorithm inserts vector points to fit the geometry of the corresponding geographic feature on the map. The number of vector points has a positive correlation to the algorithm processing time. For example, if the vector data after the densification process have ten vector points, and each vector point has nine movements, the total number of movement combinations is 10^9. If the vector data have five vector points, the total number of movements combinations reduces to 5^9. Therefore, our algorithm takes a shorter time to find the solution to alignment in the 5-vector-point case than the 10-vector-point case. The second factor is the map dimension. The map dimension determines the size of the geographic feature given the same coverage area. For example, the same railroad segment is shorter in a small dimension map than a large dimension one. After the densification process, fewer vector points are needed for the same railroad segment in a small dimension map than in a large dimension one. Hence, our algorithm is more efficient for small dimension maps than large dimension ones. The third factor is the number of movements. Our algorithm defines nine movements for each vector point. If we reduce the number of movements to five, the number of movement combinations also reduces, which leads to a reduced search space. The smaller the search space, the shorter the processing time of our algorithm.

5. Experiment and analysis

This section first describes the geographic features and maps used in the experiment and then introduces the experiment settings and evaluation metrics. After that, the section presents the experiment results and accuracy assessment.

5.1. Experiment data

We tested the proposed alignment algorithm for three geographic features (railroads, water lines, and roads) on three scanned historical maps. These three maps cover Bray, California (circa 2001), Louisville, Colorado (circa 1965), and Boulder, Colorado (circa 1966), respectively. The map scale of all test maps is 1:24,000. Each pixel on the test maps represents around one square meter. We tested the alignment of water lines in the Bray, Louisville, and Boulder maps, the alignment of railroads in the Bray and Louisville maps, and the alignment of roads in the Bray map. We downloaded the historical maps as TIFF images from the USGS topographic historical map archive (https://nationalmap.gov/) and contemporary vector data as ESRI shapefiles from the US Census Bureau website (https://www.census.gov/) for railroads and the USGS website (https://nationalmap.gov/) for water lines and roads.

The test geographic features are represented by graphical symbols of a variety of geometric characteristics. Figure 11 shows the symbols for railroads, water lines, and roads in the test maps. Railroads are represented as black lines with crosses (Figure 11(a)). Water lines are represented as blue lines of three kinds of symbols indicating different types of hydrographic features. They are solid blue lines (Figure 11(b)), solid blue lines with blue dots between them (Figure 11(c)), and blue dashed lines (Figure 11(d)). There are multiple symbols for different categories of roads as well (Figure 11(e–g)).

Besides the diverse geographic feature representations, each test map covers a large area (i.e. a city in the US) and contains rich and diverse graphical conditions. Figure 12 shows the graphically diverse neighborhoods around railroad features on the Bray map, including elevation contours, gravel roads, regular roads, and wetlands. The varying representations of geographic features and graphical conditions in these maps represent the degree of heterogeneity in the USGS topographic historical map archive. In the next subsections, we will describe the evaluation metrics and evaluate the performance of our algorithm applied to various geographic feature representations and graphical neighborhoods.

5.2. *Experiment setting and evaluation metrics*

This section discusses the hyperparameters used in the experiment and the evaluation metrics. For densification of the vector polylines, our algorithm added points every 30 pixels for railroads and roads and every 15 pixels for water lines, because there are more curves in water lines than railroads and roads, as explained before. After that, our algorithm used a 50*50-pixel window to crop sub-images from the input map to find possible movement for each vector point (including the interpolated points).

We used correctness and completeness (Heipke *et al*. 1997) as the metrics to evaluate the alignment results. Correctness is the length of true positives divided by the sum of the length of true positives and false positives, and completeness is the length of true positives divided by the sum of the length of true positives and false negatives. Figure 13 illustrates the correctness (top) and completeness (bottom). Correctness shows the percentage of the correctly aligned line features with the geographic feature of interest on the map. Completeness shows how much of the geographic feature of interest on the map correctly aligns with the repositioned vector data. We calculated the two measures for various buffer sizes to accommodate for varying symbol widths of the geographic feature of interest. We manually aligned the vector data to the centerlines of the geographic feature of interest on the maps to create reference data for our evaluation.[4]

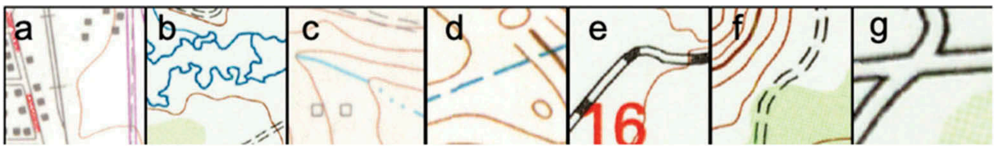

Figure 11. Examples of geographic features symbols in the test maps: (a) railroads, (b)-(d) water lines, and (e)-(g) roads.

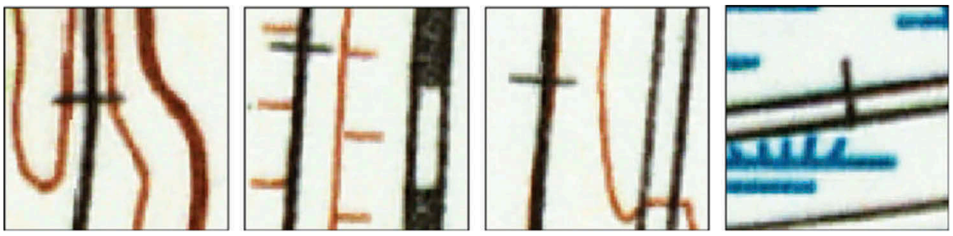

Figure 12. Examples of graphically diverse neighborhoods around railroads in the Bray map.

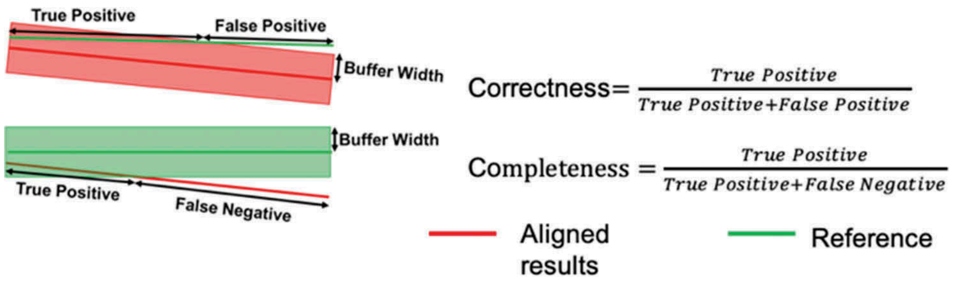
True Positive
False Positive
Buffer Width
Buffer Width
True Positive
False Negative
Correctness= True Positive / True Positive+False Positive
Completeness = True Positive / True Positive+False Negative
Aligned results
Reference

Figure 13. The illustration of correctness and completeness (the red line is the aligned result, and the green line is the reference.).

5.3. Experiment results and analysis

Tables 2–4 show the alignment results for railroads, water lines, and roads in the test maps compared to the contemporary vector data. The aligned vector data ('aligned') refer to the vector data processed by our alignment algorithm, and the contemporary vector data ('original') refer to the vector data downloaded from the US Census and USGS websites. Figure 14 shows example results for railroads, water lines, and roads over large areas. In general, we can see that our algorithm could handle various misalignment cases and geographic feature symbols. For example, our algorithm could handle the misalignment caused by projection-related distortion (Figure 14(a)), the misalignment caused by existing water lines that do not exist in the map by removing them (Figure 14(b)), and the

Table 2. The alignment results for railroads compared with the contemporary (original) vector data (the width of the railroads is 5 pixels and 3 pixels in the Bray and Louisville maps, respectively).

		Buffer (pixels)	1	3	5	7
Bray	Correctness	Original	5.32%	16.17%	**29.88%**	46.03%
		Aligned	36.53%	86.57%	**97.32%**	98.83%
	Completeness	Original	4.38%	16.39%	**29.92%**	41.92%
		Aligned	35.90%	89.42%	**90.86%**	90.52%
Louisville	Correctness	Original	7.25%	**24.81%**	39.85%	58.18%
		Aligned	42.21%	**96.57%**	97.48%	98.04%
	Completeness	Original	4.70%	**12.71%**	21.84%	35.90%
		Aligned	41.12%	**95.05%**	95.05%	94.52%

Table 3. The alignment results for water lines compared with the contemporary (original) vector data (the width of water lines is 5 pixels in all three maps).

		Buffer (pixels)	1	3	5	7
Bray	Correctness	Original	44.05%	56.46%	**62.02%**	66.28%
		Aligned	45.35%	71.69%	**84.89%**	90.30%
	Completeness	Original	55.55%	79.55%	**88.24%**	91.82%
		Aligned	46.10%	81.16%	**93.31%**	89.88%
Louisville	Correctness	Original	10.62%	23.28%	**37.01%**	50.31%
		Aligned	24.80%	68.77%	**84.58%**	90.26%
	Completeness	Original	15.36%	33.53%	**53.12%**	71.67%
		Aligned	24.99%	67.03%	**81.79%**	87.26%
Boulder	Correctness	Original	26.86%	42.30%	**59.57%**	72.47%
		Aligned	25.36%	66.04%	**81.64%**	87.45%
	Completeness	Original	32.29%	50.85%	**71.49%**	86.84%
		Aligned	26.21%	65.89%	**80.77%**	86.59%

Table 4. The alignment results for roads compared with the contemporary (original) vector data (the width of roads is 15 pixels in the Bray map).

	Buffer (pixels)	1	3	5	7	9	11	13	15
Correctness	Original	41.11%	49.11%	58.32%	67.29%	73.89%	78.23%	80.88%	**82.70%**
	Aligned	5.30%	14.14%	23.59%	37.51%	58.32%	77.62%	87.57%	**91.25%**
Completeness	Original	42.08%	51.39%	62.74%	74.21%	82.77%	87.95%	91.02%	**93.08%**
	Aligned	5.34%	13.57%	22.46%	36.30%	57.07%	75.49%	84.58%	**88.00%**

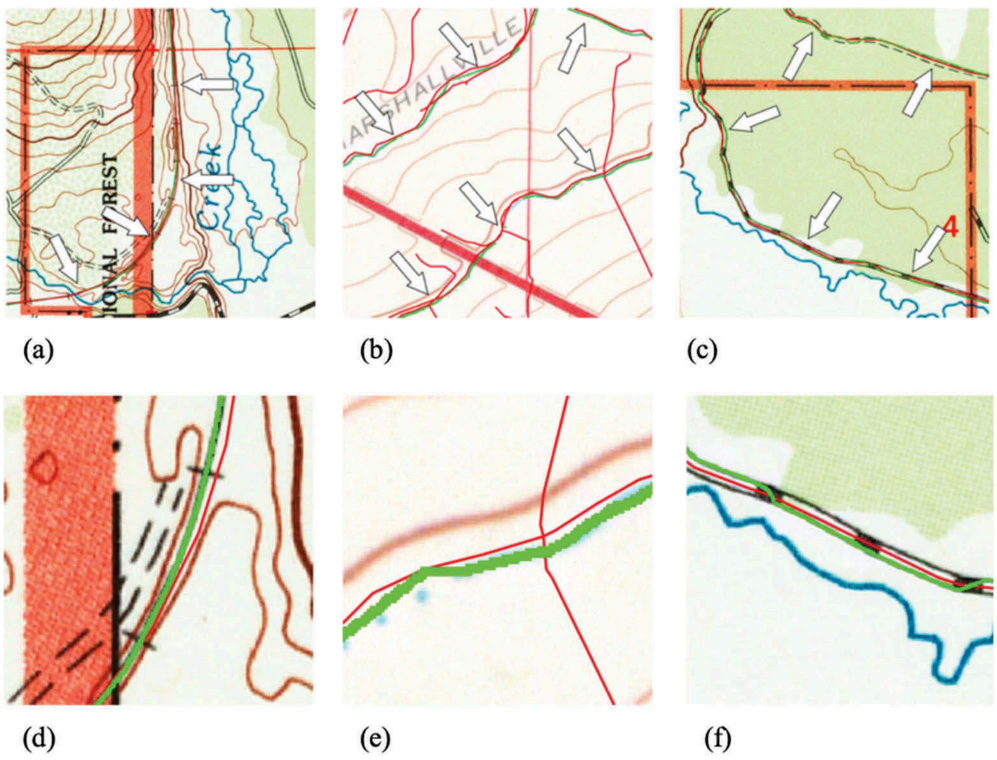

Figure 14. The sub-maps from (a) to (c) show the alignment for railroads in the Bray map, water lines in the Louisville map, and roads in the Bray map. The red lines are original vector lines, and the green lines are aligned vector lines. From (a) to (c), the white arrows point out the locations of geographic features on the maps. From (d) to (f) show the corresponding enlarged portions for each geographic feature from (a) to (c) correspondingly.

misalignment caused by the road representation inconsistency by aligning vector data to the road edge (Figure 14(c)). Our alignment algorithm also handled complex graphical conditions surrounding the target geographic feature. For example, in Figure 14(a), the railroads overlap with water lines and the red features on the map, and our algorithm correctly aligned the railroad vector data to the railroads on the map. Similarly, in Figure 14(b), our algorithm aligned the water-lines vector data to the water lines on the map, although water lines on the map overlap with other geographic features.

Table 2 shows that our algorithm aligned most of the railroad vector data to the railroads on both the Bray and Louisville maps. In the Bray map, 97.32% of the vector data were correctly aligned to the railroads within a 5-pixel buffer, the width of railroads. The correctness and completeness of the aligned vector data are about three times higher than for the original vector data. Although our algorithm does not guarantee to align the vector data to the center lines of the target feature, our algorithm still got 36.53% correctness when the buffer size is 1 pixel (i.e. the center lines of railroads), compared to 5.32% alignment of the original vector data. Figure 15 shows an example of the aligned vector data (green lines) in comparison to the reference data (red lines) with various buffer sizes in the Bray map. Note the length of the green lines contained in the buffer (i.e. correctly aligned vector segments) does not increase much with increasing buffer size from 5 to 7 pixels since the width of the map feature is approximately 5 pixels. Figure 16(a) shows an example of the vector data in comparison to the railroad features in the map before and after using our alignment algorithm in the Bray map. In the Louisville map, 96.57% aligned railroad vector data is within a 3-pixel buffer, the width of railroads, and 42.21% aligned vector data is at the center lines. Figure 16(b) shows an example of the vector data in comparison to the railroad features in the map before and after using our alignment algorithm in the Louisville map.

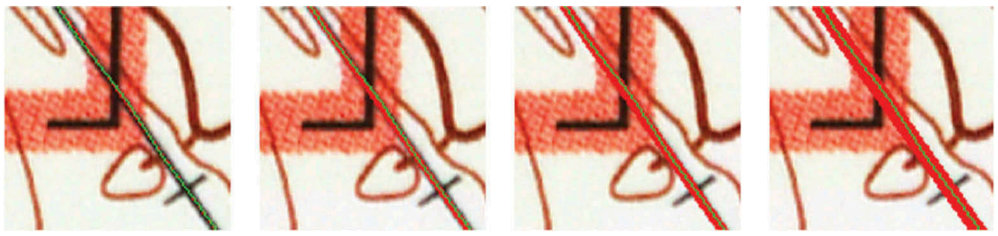

Figure 15. An example of the aligned railroad vector data (the green lines) compared with the reference data (the red lines) in various buffer sizes in the Bray map. From left to right, the buffer size is 1, 3, 5, and 7 pixels.

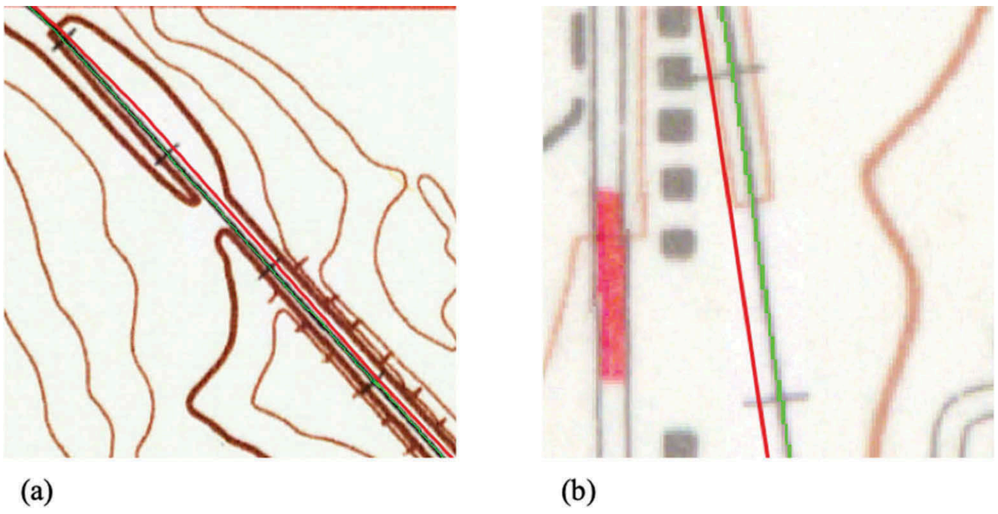

Figure 16. Examples that show the railroad alignment results. The red line is the original railroad vector, and the green line is the rasterized aligned railroad vector line processed by our algorithm. (a) is the Bray map, and (b) is the Louisville map. The left bottom in (a) is a zoom-in view of the alignment results.

Table 3 shows the alignment results for water lines, our algorithm achieved high performance for all three test maps. In the Bray map, our algorithm aligned 84.89% of the water-lines vector data within the 5-pixel buffer, the width of water lines in the map. The correctness of the aligned vector data is about 30% higher than the original vector data. Figure 17 shows an example of the aligned water-lines vector data (green lines) in comparison to the reference (red lines) with various buffer sizes in the Bray map. In the Louisville map, the correctness of the aligned water-lines vector data is about two times higher than the original vector data when the buffer is 5 pixels, the width of water lines. In the Boulder map, the correctness of the aligned water-lines vector data is 81.64% compared 59.57% for the original vector data when the buffer is 5-pixel, the width of water lines.

In Table 4, the correctness of aligned road vector data achieved 91.25%, while the correctness of original vector data is 82.7% when the buffer size is 15 pixels, the width of roads in the Bray map. However, the completeness of the aligned vector data is slightly lower than the original vector data when the buffer is 15 pixels because our alignment algorithm deleted a small portion of the original vector data far away from the roads in the map. Figure 2(c) shows an example of this kind of road segment. Since our algorithm assumes that the corresponding features from the two datasets should be close to each other, if the position offset between them is significant, the algorithm considers them as different features. As for aligning to the center lines, we can see from Table 4 that the correctness and completeness of the aligned vector data are only 5.30% and 5.34%, respectively, when the buffer size is 1 pixel, while they are 41.11% and 42.08%, respectively, for the original vector data. This is because the road line symbol is not a homogenous color area. For example, the center of the road lines contains both black pixels and background pixels (filled and non-filled) shown in Figure 11(e–g), so our algorithm would align the vector data to the road edges, which have a consistent road representation.

5.4. Limitations and considerations

This section discusses the current limitations and practical considerations of the proposed algorithm.

5.4.1. Spatial relationship (parallel and intersection)

Our algorithm could take spatial and topological relationships (e.g. parallel railroads in Figure 18(a)) into account during the alignment process to improve the performance. For example, our algorithm aligned the parallel vector lines to the same railroad on the Bray map. Some of the railroad tracks in the Louisville map are also represented by parallel

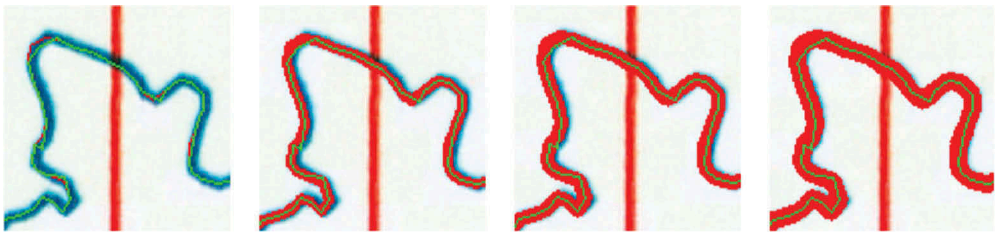

Figure 17. The example of the aligned water-lines vector data (the green lines) compared with the reference (the red lines) in various buffer sizes in the Bray map. From left to right, the buffer size is 1, 3, 5, and 7 pixels.

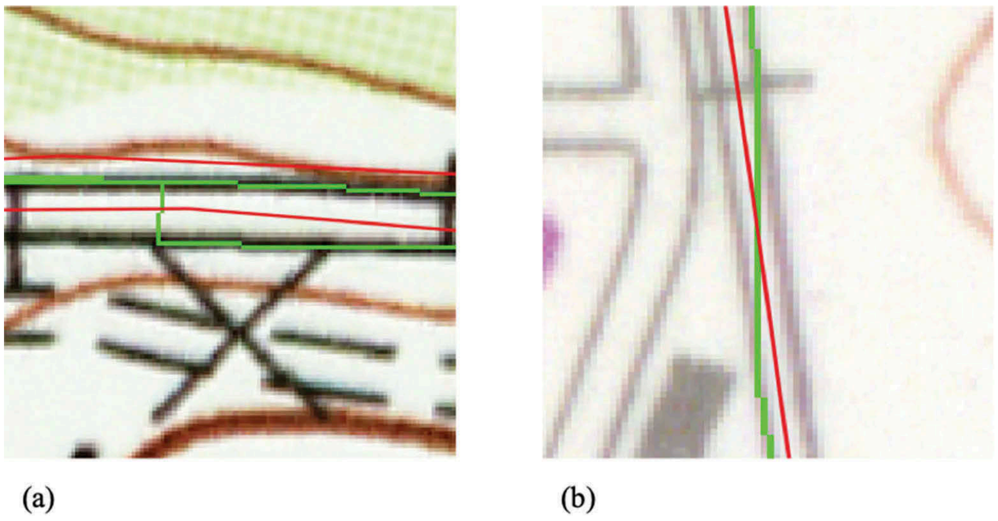

Figure 18. Examples that show the alignment results on parallel railroads. The red line is the contemporary railroad vector, and the green line is the aligned railroad vector processed by our algorithm. (a) and (b) is an example for the Bray and Louisville map respectively.

lines, but only one track exists in the vector data (Figure 18(b)). Our algorithm aligned the single railroad vector line to the parallel railroad lines because the vector polyline consists of several segments, and our algorithm did not process the segments at the same time.

Processing each line segment one by one also could result in losing geometric and topological information of intersections. Figure 19 shows examples of road intersections. In Figure 19(a), two intersecting road vector lines moved in two opposite directions to align to the edges of roads to have the maximum rewards resulting in the loss of the

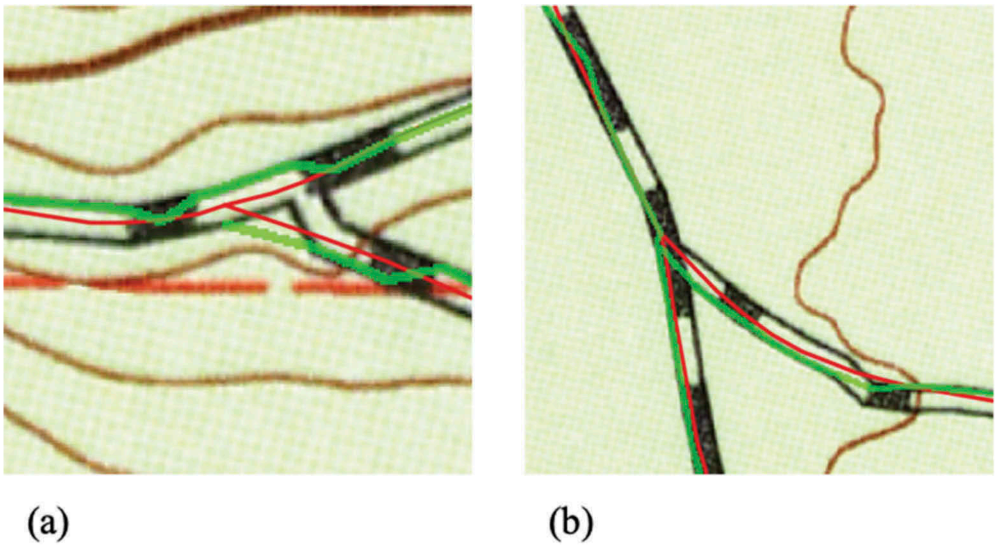

Figure 19. (a) is an example of the loss of road intersections, and (b) is an example of the distortion of intersections. The red lines are the original vector data. The green lines are the vector data processed by our algorithm.

intersection. Figure 19(b) shows an example in which individual alignment can distort the road angle. Our algorithm aligned each vector line to the edges of the roads on the map to get the maximum reward but did not consider them as a group. A possible solution is to consider the spatial or topological relationships of geographic features (vector data) in the reward function to maintain them during the process.

5.4.2. Symbol representation

Our algorithm could not perform well when the map symbol does not have a consistent color. For example, water lines and roads are represented by dashed lines on maps in Figure 11(c–f), respectively. Another example is the road represented by the filled and non-filled symbols or hollow symbols in Figure 11(e,g), respectively. For the dashed lines, the vector line could not receive the maximum reward because some segments would always align with the background. Figure 20 shows some examples of the alignment performance for dashed water lines. Because water lines are represented as solid blue lines in the Bray map but dashed and dotted lines in the Louisville and Boulder maps, the correctness and completeness of water-line alignment in the Louisville and Boulder map are slightly lower than that in the Bray map. In the case of road features, Figure 21 shows that the road vector data often align to the edge of roads in the maps instead of the center.

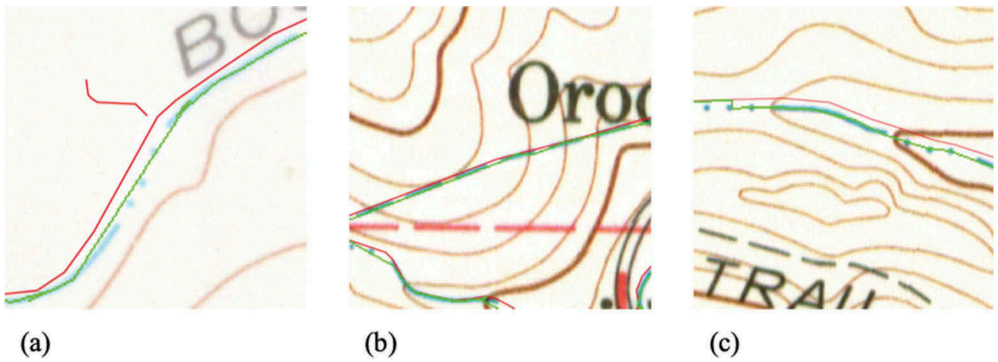

Figure 20. Examples that show water-lines vector data aligning to the dashed and dotted water lines in the maps. The red lines are the original water-lines vector data and the green lines are the aligned vector data. (a) is an example for the Louisville map, and (b), (c) are for the Boulder map.

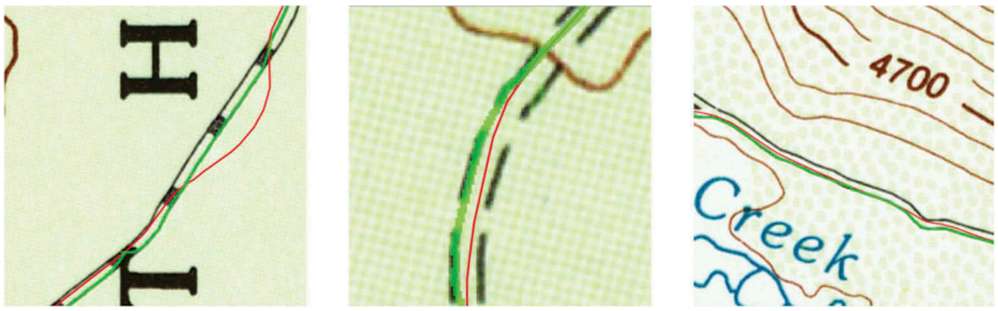

Figure 21. Road alignment results in the Bray map with various road symbol representations. The red line is the original road vector, and the green line is the aligned road vector processed by our algorithm.

5.4.3. Geometric representation

The geometric representation of the same geographic feature may not be similar in the vector data and map. For example, the water lines are represented as curves in the map while they are straight lines in the vector data in some locations (Figure 22). Our algorithm densified the water-line vector data by interpolating points every 15 pixels to solve this problem. However, the densification can accommodate the wide curves well but not the tight ones with sudden changes (Figure 23). Adding more vector points during the densification process would improve the alignment process for such tight curves.

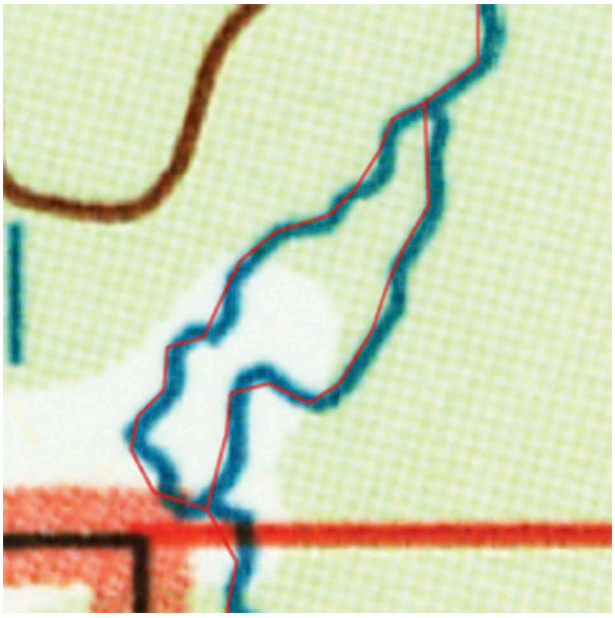

Figure 22. An example of different feature representations for the water lines in contemporary vector data and the Bray map. Water lines are represented as straight line segments in the vector data (the red lines), while curves in the map.

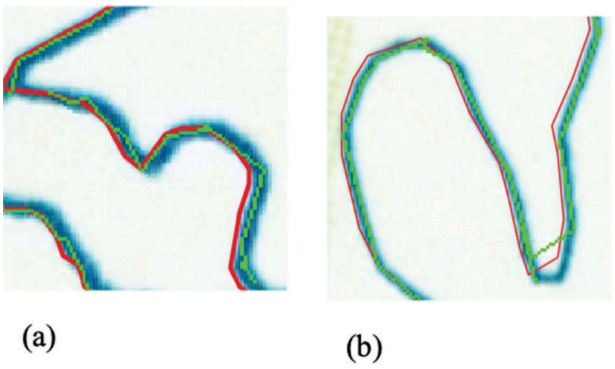

Figure 23. An example shows how our algorithm can align the original vector data (the red lines) to the water lines (the blue lines) with wide curves (a) but not the tight curves (b) on the map. The green lines are the (rasterized) aligned vector data.

5.4.4. Large positional offsets

Our algorithm uses the position offset as a hint to verify if the content on the map and the auxiliary vector data represent the same feature. If a nearby geographic feature can be found in the map in at least one of cardinal or inter-cardinal directions, this feature is considered the corresponding one. Otherwise, the algorithm removes the vector features from the process. Having a large positional offset often means that the geographic features from the two data sources are significantly different (e.g. the changes of road networks from maps circa 1895 and 2018), which would be a challenge for any alignment algorithm.

5.4.5. Reward function

When overlapping non-target features have a similar color, these non-target features can be falsely identified as the target feature, and hence our reward function would give positive rewards to mis-aligned segments. Figure 24 shows an example where the contour line connects to the railroads in one area of the Bray map. The green line represents the vector data aligned by our algorithm. In this example, first, our algorithm moves the vector points to the contour lines due to the similar color range of the railroads and contour lines. Next, because of the similar color range and the fact that the railroads and contour lines are connected, the reward function gives positive rewards to these incorrect movements. This limitation could be solved by improving the process of identifying the color range of the target feature so that our algorithm can better separate the target feature from other features. Also, the reward function could take into account expected geometric properties due to the semantics of the target features (e.g. railroads should not have a sharp turn) and penalize unlikely feature geometry in the alignment results.

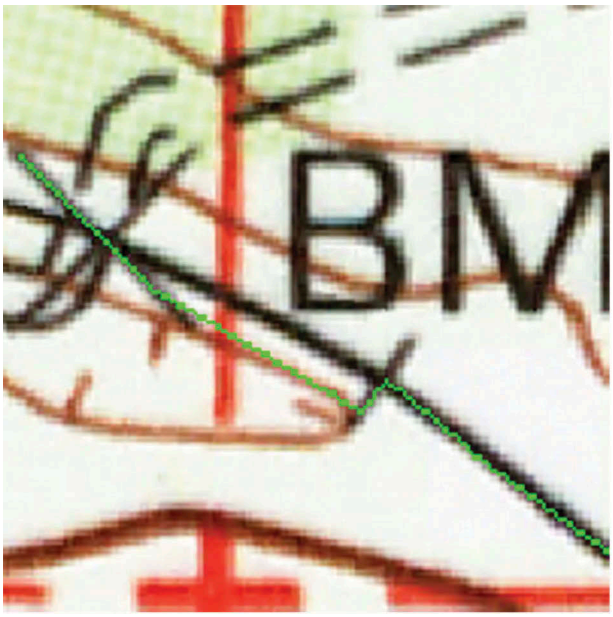

Figure 24. An example of the mis-aligned railroads where the green line is the output vector data from our algorithm.

6. Conclusion and future work

In this paper, we presented a vector-to-raster alignment algorithm based on the reinforcement learning framework, which makes the alignment process efficient for various types of geographic features without human intervention. In the experiment section, we showed that our alignment algorithm achieved accurate results in most cases. Automatically aligning the vector data to the geographic feature of interest in the scanned historical maps can help to automatically generate a large number of representative samples of a map symbol necessary for training stages in machine learning frameworks for information extraction (e.g. Convolutional Neural Networks) to recognize geographic features in digital maps. Recognizing the geographic features in historical maps can convert valuable information locked in those maps into machine-readable datasets.

Future work will focus on further improvement of our algorithm in two aspects. First, we plan to improve the reward function to handle a variety of symbols (dashed and double lines), maintain the spatial and topological relationships between and within geographic layers in vector format, and avoid the misalignment that violates semantic properties of geographic features. Second, we plan to build the technology to accommodate varying feature representations (e.g. from map generalization) to apply the algorithm to maps of different scales.

Notes

1. https://cs.stanford.edu/~roozbeh/pascal-context/.
2. This method is based on our previous work published in the first GeoAI Workshop (Duan *et al.* 2017a).
3. However, the poor quality of the original scanned maps may violate the assumption that the color for the geographic feature is consistent and distinguishable and may have a negative impact on the accuracy of our algorithm. For example, when the original maps are prone to color bleaching (Leyk and Boesch 2010, Chiang *et al.* 2014), the color cannot be used as a distinct property for the geographic feature. As a consequence, our reward function, which is designed based on the color property, cannot provide accurate rewards to actions. The color property should be valid and consistent for most of the scanned maps edited in recent decades.
4. The manually aligned vector data are reviewed by researchers who are not on the author list of this paper but are affiliated with the same research institute.

Acknowledgments

This material is based on research sponsored in part by the National Science Foundation under Grant Nos. IIS 1563933 (to the University of Colorado at Boulder) and IIS 1564164 (to the University of Southern California).

Disclosure statement

No potential conflict of interest was reported by the authors.

Data and codes availability statement

The source code and data that support the findings in this study are available on figshare. The source code of the vector-to-raster alignment algorithm is available on https://doi.org/10.6084/m9.figshare.10025051.v1. For the data, the contemporary vector data and the USGS topographic maps are available on https://doi.org/10.6084/m9.figshare.9999863.v1 and found on https://doi.org/10.6084/m9.figshare.9999740.v1, respectively.

Funding

This work was supported by the National Science Foundation [1563933,1564164].

Notes on contributors

Weiwei Duan is a Ph.D. student majoring in Computer Science at the University of Southern California (USC). She is working on building a computer-vision-based system for extracting information on georeferenced images and storing them in a structured format for analysis. The system localizes geographic objects on images by integrating geospatial information and using limited noisy labeling data. Her research interests are computer vision, knowledge graphs, and machine learning.

Yao-Yi Chiang, Ph.D., is an Associate Professor (Research) in Spatial Sciences, the Director of the Spatial Computing Laboratory, and the Associate Director of the NSF's Integrated Media Systems Center (IMSC) at the University of Southern California (USC). He is also a faculty in Data Science in the USC Viterbi Data Science M.S. program. Dr. Chiang received his Ph.D. degree in Computer Science from the University of Southern California; his bachelor's degree in Information Management from the National Taiwan University. His current research combines spatial science theories with computer algorithms to enable the discovery of useful insights from heterogeneous data for solving real-world problems. His research interests include information integration, machine learning, data mining, computer vision, and knowledge graphs. Before USC, Dr. Chiang worked as a research scientist for Geosemble Technologies and Fetch Technologies in California. Geosemble Technologies was founded based on a patent on geospatial data fusion techniques, and he was a co-inventor.

Stefan Leyk is an Associate Professor at the Department of Geography, University of Colorado Boulder and a Research Fellow at the Institute of Behavioral Science. He is a Geographical Information Scientist with research interests in information extraction, spatio-temporal modeling and socio-environmental systems. In his work he uses various sources of historical spatial data to better understand the evolution of human systems and how the built environment interacts with environmental processes in the context of land use and natural hazards.

Johannes H. Uhl received a diploma degree in surveying and geomatics from Karlsruhe University of Applied Sciences, Germany, in 2009 and a double graduation M.Sc. degree in geomatics from Karlsruhe University of Applied Sciences, Germany and in cartography and geodesy from Polytechnic University of Valencia (UPV), Spain in 2011. He received his Ph.D. degree in geographic information science from University of Colorado, Boulder, USA in 2019.In 2008/2009, he worked as a student intern at Department of Photogrammetry and Image Analysis (IMF) at the German Aerospace Center, Oberpfaffenhofen, Germany. From 2012 to 2015, he worked as Geospatial Data Analyst and Software Developer at Pfalzwerke Netz AG, Ludwigshafen, Germany, and from 2016 to 2019 as Graduate Research Assistant at the Department of Geography, University of Colorado, Boulder, USA. Since 2019, he is a Post-Doctoral Research Associate at the University of Colorado Population Center (CUPC), Institute of Behavioral Science at University of Colorado, Boulder, USA. His research interests include the efficient integration and analysis of large geospatial datasets, spatio-temporal modeling and information extraction based on multi-source geospatial data using machine learning, image processing and statistical analysis, as well as uncertainty quantification and modeling of spatio-temporal data. Dr. Uhl was a recipient of the Best Paper Award at the International Conference of Pattern Recognition Systems (ICPRS) 2017 in Madrid, Spain and received the Gilbert F. White Fellowship from University of Colorado in 2018.

Craig A. Knoblock is Executive Director of the Information Sciences Institute of the University of Southern California (USC), Research Professor of both Computer Science and Spatial Sciences at USC, and Director of the Data Science Program at USC. He received his Bachelor of Science degree from Syracuse University and his Master's and Ph.D. from Carnegie Mellon University in computer science. His research focuses on techniques for describing, acquiring, and exploiting the semantics of data. He has worked extensively on source modeling, schema and ontology alignment, entity and record linkage, data cleaning and normalization, extracting data from the Web, and combining all of these techniques to build knowledge graphs. He has published more than 300 journal articles, book chapters, and conference papers on these topics and has received 7 best paper awards on this work. Dr. Knoblock is a Fellow of the Association for the Advancement of Artificial Intelligence (AAAI), a Fellow of the Association of Computing Machinery (ACM), past President and Trustee of the International Joint Conference on Artificial Intelligence (IJCAI), and winner of the Robert S. Engelmore Award.

ORCID

Stefan Leyk http://orcid.org/0000-0001-9180-4853
Johannes H. Uhl http://orcid.org/0000-0002-4861-5915

References

Chen, B., Sun, W., and Vodacek, A., 2014. Improving image-based characterization of road junctions, widths, and connectivity by leveraging OpenStreetMap vector map. *In*: *The IEEE Geoscience and Remote Sensing Symposium (IGARSS) proceedings*, 3 July 2014. Quebec, Canada: IEEE International, 4958–4961.

Chen, C., Knoblock, C.A., and Shahabi, C., 2006. Automatically conflating road vector data with orthoimagery. *GeoInformatica*, 10 (4), 495–530. doi:10.1007/s10707-006-0344-6.

Chen, L.C., *et al.*, 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. *In*: *Proceedings of the European Conference on Computer Vision (ECCV)*, 801–818. doi:10.1177/1753193418780184.

Chiang, Y.Y. and Knoblock, C.A., 2015. Recognizing text in raster maps. *Geoinformatica*, 19 (1), 1–27. doi:10.1007/s10707-014-0203-9.

Chiang, -Y.-Y., Leyk, S., and Knoblock, C.A., 2014. A survey of digital map processing techniques. *ACM Computing Surveys*, 47 (1), 1–44. doi:10.1145/2557423.

Duan, W., *et al*. 2017a. Automatic alignment of geographic features in contemporary vector data and historical maps. *In*: *The 1st workshop on artificial intelligence and deep learning for geographic knowledge discovery proceedings*, 7 Nov 2017. Redondo Beach: ACM, 45–54.

Duan, W. and Chiang, -Y.-Y., 2017b. SRC: a fully automatic geographic feature recognition system. *SIGSPATIAL Special*, 9 (3), 6–7. doi:10.1145/3178392.3178396.

Heipke, C., *et al.*, 1997. Evaluation of automatic road extraction. *International Archives of Photogrammetry and Remote Sensing*, 32 (3 SECT 4W2), 151–160.

Kaelbling, L.P., Littman, M.L., and Moore, A.W., 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 237–285. doi:10.1613/jair.301

Kober, J., Bagnell, J.A., and Peters, J., 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32 (11), 1238–1274. doi:10.1177/0278364913495721.

Krizhevsky, A., Sutskever, I., and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1106 – 1114.

Le Moigne, J., Campbell, W.J., and Cromp, R.F., 2002. An automated parallel image registration technique based on the correlation of wavelet features. *IEEE Transactions on Geoscience and Remote Sensing*, 40 (8), 1849–1864. doi:10.1109/TGRS.2002.802501.

Leyk, S. and Boesch, R., 2010. Colors of the past: color image segmentation in historical topographic maps based on homogeneity. *GeoInformatica*, 14 (1), 1–21. doi:10.1007/s10707-008-0074-z.

Osmanković, D. and Konjicija, S., 2011. Implementation of Q—learning algorithm for solving maze problem. *In*: *The 34th international convention proceedings MIPRO*. Opatija, Croatia: IEEE, 1619–1622.

Ostafin, K., *et al*., 2017. Forest cover mask from historical topographic maps based on image processing. *Geoscience Data Journal*, 4 (1), 29–39. doi:10.1002/gdj3.46.

Razavian, A.S., *et al*., 2014. CNN features off-the-shelf: an astounding baseline for recognition. *arXiv preprint arXiv:1403.6382*.

Rother, C., Kolmogorov, V., and Blake, A., 2004. Grabcut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)* ACM, 23 (3), 309–314. doi:10.1145/1015706.

Ruiz-Lendínez, J.J., *et al*., 2017. Method for an automatic alignment of imagery and vector data applied to cadastral information in Poland. *Survey Review*, 51 (365), 123–134. doi:10.1080/00396265.2017.1388959.

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, W., *et al*., 2013. An automated approach for the conflation of vector parcel map with imagery. *Photogrammetric Engineering & Remote Sensing*, 79 (6), 535–543. doi:10.14358/PERS.79.6.535.

Sutton, R.S., 1996. Generalization in reinforcement learning: successful examples using sparse coarse coding. *In*: *Advances in neural information processing systems*, Cambridge, MA: MIT press, 1038–1044.

Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: an introduction*. Cambridge, MA: MIT press.

Tong, X., Shi, W., and Deng, S., 2009. A probability-based multi-measure feature matching method in map conflation. *International Journal of Remote Sensing*, 30 (20), 5453–5472. doi:10.1080/01431160903130986.

Touya, G., *et al*., 2018. Multi-agents systems for cartographic generalization: feedback from past and on-going research. Doctoral dissertation. IGN (Institut National de l'Information Géographique et Forestière).

Uhl, J.H., *et al*. 2017. Extracting human settlement footprint from historical topographic map series using context-based machine learning. *In*: *The 8th international conference on pattern recognition systems proceedings*, Madrid, Spain: IET, 1–6.

Uhl, J.H., *et al*., 2018a. Spatialising uncertainty in image segmentation using weakly supervised convolutional neural networks: a case study from historical map processing. *IET Image Processing*, 12 (11), 2084–2091. doi:10.1049/iet-ipr.2018.5484.

Uhl, J.H., *et al*., 2018b. Map archive mining: visual-analytical approaches to explore large historical map collections. *ISPRS International Journal of Geo-Information*, 7 (4), 148. doi:10.3390/ijgi7040148.

Watkins, C.J. and Dayan, P., 1992. Q-learning. *Machine Learning*, 8 (3–4), 279–292. doi:10.1007/BF00992698.

Wu, X., 2011. *Method for automatic alignment of raster data with vector data in a geographic information system*. US Patent No. 7,869,667.

Wu, X., *et al*., 2007. Automatic alignment of large-scale aerial rasters to road-maps. *In*: *The 15th annual ACM international symposium on Advances in geographic information systems proceedings*, 7 Nov 2007. Seattle, USA: ACM, 17.