MI3: Machine-Initiated Intelligent Interaction for Interactive Classification and Data Reconstruction

YU ZHANG, University of Oxford, The United Kingdom
BOB COECKE, University of Oxford, The United Kingdom
MIN CHEN, University of Oxford, The United Kingdom

In many applications, while machine learning (ML) can be used to derive algorithmic models to aid decision processes, it is often difficult to learn a precise model when the number of similar data points is limited. One example of such applications is data reconstruction from historical visualizations, many of which encode precious data, but their numerical records are lost. On the one hand, there is not enough similar data for training an ML model. On the other hand, manual reconstruction of the data is both tedious and arduous. Hence, a desirable approach is to train an ML model dynamically using interactive classification, and hopefully, after some training, the model can complete the data reconstruction tasks with less human interference. In order for this approach to be effective, the number of annotated data objects used for training the ML model should be as small as possible, while the number of data objects to be reconstructed automatically should be as large as possible. In this paper, we present a novel technique for the machine to initiate intelligent interactions to reduce the user's interaction cost in interactive classification tasks. The technique of machine-initiated intelligent interaction (MI3) builds on a generic framework featuring active sampling and default labeling. To demonstrate the MI3 approach, we use the well-known cholera map visualization by John Snow as an example as it features three instances of MI3 pipelines. The experiment has confirmed the merits of the MI3 approach.

CCS Concepts: • **Human-centered computing** → **Interactive systems and tools**; *Visualization*; • **Theory of computation** → **Active learning**; • **Applied computing** → *Data recovery*; • **Computing methodologies** → Object identification.

Additional Key Words and Phrases: Intelligent user interface, active learning, interactive classification, data labeling, interaction reduction, data reconstruction, historical visualization

ACM Reference Format:
Yu Zhang, Bob Coecke, and Min Chen. 2021. MI3: Machine-Initiated Intelligent Interaction for Interactive Classification and Data Reconstruction. *ACM Trans. Interact. Intell. Syst.* 11, 3-4, Article 18 (August 2021), 34 pages. https://doi.org/10.1145/3412848

1 INTRODUCTION

History has left us many wonderful visualization images. The map collection by David Rumsey alone includes about 3,000 statistical graphics images and thematic maps, in addition to some 100,000 geographical maps [66]. There are many such collections around the world. Many historical visualizations capture important statistics of historical events, and therefore are of great interest to scholars in humanities and social sciences. In numerous cases, the original datasets are lost. It is

Authors' addresses: Yu Zhang, University of Oxford, Oxford, OX1 3QD, The United Kingdom, yu.zhang@cs.ox.ac.uk; Bob Coecke, University of Oxford, Oxford, OX1 3QD, The United Kingdom, bob.coecke@cs.ox.ac.uk; Min Chen, University of Oxford, Oxford, OX1 3QD, The United Kingdom, min.chen@eng.ox.ac.uk.

2160-6455/2021/8-ART18 $15.00
https://doi.org/10.1145/3412848

desirable to reconstruct the datasets from the visualization images. Some of the most well-known historical visualizations include:

- William Playfair's time series chart of trade balance (1786) [58], bar chart of Scotland's imports/exports (1786) [58], and pie chart of Turkish Empire's land holdings (1801) [60];
- Charles Joseph Minard's flow map of road traffic (1845) [51], and flow map of Napoleon's Russian campaign (1869) [53];
- John Snow's cholera map (1854) [74, 75];
- Florence Nightingale's coxcomb chart (1858) [55, 56].

Reconstructing a dataset manually with pen-and-ruler is laborious. Many attempts have been made to extract data from computer-generated visualization images. Some [25, 70, 88] used hand-crafted image processing algorithms, while others [2, 33, 38, 48, 62, 63, 73] used machine learning to derive models for recognizing visual objects that encode data.

However, unlike computer-generated imagery, a historical visualization image typically features a unique visual design. In addition, the hand-drawn nature of the visualization and the deterioration of and the damages to the paper media pose further challenges to the process of recovering the data from historical visualization images. If there were a fully-automated solution, one would have to develop an algorithm for each image individually by programming or using machine learning. Likely the algorithm would be unsuitable for other historical visualization images. Therefore, the cost of manually recovering data would likely be lower than programming, and ironically would be almost the same as the cost of preparing a training dataset for a machine learning process.

A practically more effective solution would be an interactive intelligent system that is equipped with a collection of elementary algorithms, asks users questions intelligently, and adapts its algorithms automatically to work with a given historical visualization image. It is not an idealized system that could recover data automatically from many different historical visualization images, but a computerized assistant that can initiate intelligent interactions with human users in order to adapt itself to each specific task. The design goal of "intelligent interactions" is to minimize the number of interactions.

In this paper, we present a generic approach for machine-initiated intelligent interaction (MI3). The approach is governed by an iterative machine learning framework that features *algorithmic sampling (active sampling)* for dynamically acquiring labels from the user and *algorithmic default labeling (label propagation)* for maximizing the informational value of the user's inputs. It consists of a collection of image processing algorithms to accommodate the needs for detecting different types of visual objects. The goal of the MI3 approach is to perform the data reconstruction task with as few interactions as possible. Using machine learning to train a model is primarily for supporting the task at hand, rather than for deriving a model that can be reused for many other visualization images, since there are seldom any similar images. The main contributions of this work include:

- a generic approach for machine-initiated intelligent interaction (MI3) in the context of recovering data from historical visualization images;
- a demonstration of actualizing the MI3 approach in three functional pipelines;
- a quantitative evaluation of the effectiveness of the MI3 approach in the three pipelines;
- a prototype system that supports data reconstruction from spatial data visualization;
- an analysis of the reusability of software pipelines developed using the MI3 approach.

2 RELATED WORK

2.1 Chart Data Reconstruction

Many valuable datasets are only available through their visualization imagery in printed or scanned media. Manually reconstructing a dataset from a visualization image with pen-and-ruler is accurate but laborious, and is often not scalable for charts with many data objects, such as John Snow's cholera map [74]. It has attracted research interest in the HCI and image processing communities to reduce human effort in the data reconstruction process.

Several tools [8, 27, 80] provide digital extensions of the pen-and-ruler approach by allowing users to interactively inform the computer about where the visual objects containing the data to be recovered are. For example, Ycasd [27] is a tool for digitizing line charts. It requires the user to indicate the locations of axes, specify the axes scales, and point out individual data points. The total number of interactions required is thus at the same scale as the number of digitized data points. We will compare our MI3 approach with the computerized pen-and-ruler approach in Section 6.

Many techniques [25, 65, 70, 88] have been proposed to process visualization images automatically, and reconstruct data records from recognized data objects. Each of these techniques is constructed based on the known visual specification of particular types of charts. There are often human-controllable parameters for ensuring a good match between the techniques and minor variations of the charts within the same chart type group. Such techniques are suitable for those commonly-used and computer-generated statistical charts. However, they rarely work well with historical visualization images, typically hindered by the non-standard visual design, the distortion of the hand-drawn shapes, and the noise due to deterioration of the paper media.

In order to ensure high-quality data reconstruction, semi-automatic image processing techniques have been deployed in some systems [37, 50]. iVoLVER [50] is such an interactive system, with which the user specifies how visual objects map to data. ChartSense [37] is a mixed-initiative system, which enables users to determine image processing parameters. As our application concerns recovering valuable data from historical visualizations, the quality of data reconstruction is a crucial requirement. Hence, involving human users in the reconstruction process is unavoidable. While a user's specification of the mapping from visual objects to data and various image processing parameters can guarantee the accuracy of data reconstruction from historical visualizations, any trial-and-error interactions could easily incur undesirable effects, e.g., frustration, cognitive load, and time cost. We have thus designed our user interface for posing questions that users can answer easily without any explicit knowledge of the underlying algorithms for image processing or mapping specifications. Meanwhile, the user interface can reduce the users' burden in answering the questions by improving the correctness of the default answers using active learning. In other words, the user interface supports MI3.

To avoid the necessity for a precise specification of a type of chart, ML has been used to construct algorithmic models and define their parameters by using human-annotated datasets as the training data [2, 33, 38, 48, 62, 63, 73]. Al-Zaidy and Gile applied this approach to data reconstruction from bar charts by using decision-tree-based classifiers [2]. Poco and Heer used SVM to classify textual elements in visualization images [62]. Poco et al. further developed a technique to extract the color encoding from visualization images by recognizing the legend [63]. The combination of machine learning and image processing presents an attractive advantage when there are many annotated visualization images available for training a reasonably accurate model, and many more not-yet-annotated visualization images for which the trained model can be used to recover the unknown data. Such an advantage is absent with historical visualization images since the number of similar images available usually is insufficient as the training data. By the time all the similar visualization images are annotated for model training, there is no need for the model anymore.

This naturally leads to the idea that one may select parts of an image to train a model with possibly mediocre accuracy, and apply the trained model for the rest part of the image, and if any, some other similar images. This idea is the basis of this work. We further enhance this idea by introducing an algorithmic selection of "parts of an image" and algorithmic provision of "tentative labels" to reduce the number of interactions required for interactive classification. We will evaluate the merits of these two additions in Section 6.

2.2 Interactive Classification with Intelligent User Interfaces

Data annotation is an essential, and often costly, step for any supervised learning techniques. In several applications, intelligent user interfaces for interactive classification have been used to reduce the cost of data annotation. Fails and Olsen proposed Crayons [21], an interactive classification technique for image pixel classification, that learns the labels generated from user's paint interaction. In the area of image searching, Fogarty et al. developed a search system, CueFlik [23], with which the user can define search criteria for a concept by using positive and negative examples and rank search results according to the similarity to the concept. Active learning is used to inform the user the images that confuse the system the most, guiding the user to provide examples that benefit the concept classifier the most. To support efficient iterative image searching, Luo et al. proposed multi-class query ranking [49], a semi-supervised learning algorithm based on manifold ranking.

In the area of computer vision, Russell et al. developed LabelMe [67], a system for annotating objects in images. The user can specify an object contour by freehand drawing. Seifert and Granitzer proposed an active learning system for image labeling that allows the user to actively select data objects to be labeled with the aid of visualization [71]. Andriluka et al. presented an interactive system, Fluid Annotation [3], for annotating ground-truth results for image segmentation. It uses a neural network model trained in advance to compute an initial set of segments in an image and asks the user to edit the geometry of the segments to correct errors. In our MI3 approach, the ML framework does not assume the availability of any prelabeled training data.

Techniques for accelerating the interactive classification process in the literature can be categorized into *active-learning-based methods* and *clustering-based methods* [77]. A method in the former category attempts to select more informative data points and request the user to label them interactively. The labeled data is then used to train an interim model. With an iterative process, the method enables the interim model to be improved gradually, hence reducing the number of data points needed to be manually annotated. The latter attempts to make use of each human annotation to label more data points. In the following, we give several examples of clustering-based methods, and we will discuss the active-learning-based methods in the next subsection.

Cui et al. described an interactive photo annotation system, EasyAlbum [19], allowing the user to annotate data points in a cluster-by-cluster manner. Liu et al. described a method that first divides unlabeled data points into clusters, selects exemplars from each cluster for the user to label, and propagates the labels to other data points in the cluster [47]. Rafailidis et al. described the content-based tag propagation technique that propagates user-provided tags to similar items to address the "cold start" problem and boost the accuracy of tag-based search engine [64]. Tian et al. described a hybrid method that first groups unlabeled data points into evident clusters and a background cluster. It then allows the user to annotate each evident cluster as a whole and guides the user to annotate the background cluster using an active-learning-based method [78]. Tang et al. described a multi-scale method that allows the user to label data points in a cluster-by-cluster manner, and to refine the labels in each cluster recursively using the same clustering-based mechanism [77]. Liao et al. described an active learning system for video annotation using a semi-supervised projection algorithm in conjunction with a scatter plot and iso-contour depiction of label uncertainty [45].

In addition to images and videos, interactive classification interfaces have also been used for other applications, such as text classification, audio processing, and recommendation systems. Gervasio et al. [26] and Weber and Pollack [83] used active learning techniques to learn users' scheduling preferences. Their method balances learning efficiency and user satisfaction. Bryan et al. developed the ISSE system for source separation in audio datasets based on interactive machine learning [10].

The user can perform the separation through painting and segmentation in the time-frequency visualization of the audio record. Kucher described an interactive classification technique, ALVA, for annotating text data with the aid of visualization [41]. Active learning has been used in intelligent user interfaces for recommendation, searching, and ranking tasks. Heimerl et al. presented a visual analytics system for text document classification based on active learning [30]. They evaluated three variants of the interface for building classifiers with different levels of user control and visualization integration. Kveton and Berkovsky proposed the generalized linear search technique for providing users with algorithm-generated recommendations to reduce the interaction cost for search activities [42].

The generic MI3 approach includes an algorithmic default labeling component, which in principle can be an algorithmic clustering-based method, an interactive clustering method, or any other future method for label propagation.

2.3 Active Learning and Semi-Supervised Learning

Active learning is a family of ML methods that interact with the user during a learning process, typically (in a narrow definition) for seeking labels for unlabeled training data points, and in some cases (in a broad definition), for seeking important decisions that can improve the quality and performance of the ML process.

A critical feature of many active learning methods is to select data points "intelligently" to seek labels from the user. Lewis and Catlett proposed such a method, which estimates the uncertainty of each unlabeled data point and selects the data point with the highest uncertainty for the user to label [44]. Brinker introduced an active learning method based on diversity to sample a batch of unlabeled data points for the user to label [9]. Xu et al. described a batch method that selects data points for labeling based on their relevance, density, and diversity measures [84]. Nguyen and Smeulders described a method that uses the clustering property of the data and selects data points in favor of dense clusters [54]. Guo and Schuurmans described a batch method that selects the data points that maximize the discriminative performance of the target classifier while minimizing the entropy of the missing labels [28]. In addition, active learning has been used for accelerating interactive classification [32, 36].

The MI3 approach presented in this paper contains an active learning component. In particular, all pipelines implemented with the MI3 approach actively initiate intelligent interactions by selecting a batch of data points for the user to label. In the implementation of MI3 pipelines reported in this paper, we used the methods by Lewis and Catlett [44] and Xu et al. [84].

Semi-supervised learning is a family of ML methods that make use of both labeled and unlabeled data points for training a model. For example, Chapelle et al. proposed a framework for incorporating unlabeled data in kernel classifiers [15]. Leistner et al. extended the random forests method with a semi-supervised mechanism [43]. They reformulated the optimization goal of maximizing multi-class margin by making use of unlabeled data in addition to labeled data. Yarowsky proposed self-training [85], a boosting technique, that uses pseudo-labels of unlabeled data during training. Using a classifier learned on labeled data as an interim ML model, the technique applies the interim model to unlabeled data, and extends the training dataset that is used to retrain the model. Blum and Mitchell proposed co-training [7], a similar technique to self-training, that uses two interim classifiers trained using two disjoint subsets of the data features. The predicted tentative labels on the unlabeled dataset from one of the classifiers are fed to the other to enlarge the training set. Joachims developed the transductive support vector machine (TSVM) as an extension of the traditional SVM by involving unlabeled data in addition to labeled data for margin maximization in the model construction process [34]. Zhu and Ghahramani described a graph-based algorithm for transferring known labels to unlabeled data points iteratively [89]. Zhou et al. described another graph-based algorithm with a different iterative label spreading function inspired by spreading activation networks [87]. Tong and Jin described a mixed label propagation algorithm [79] that exploits the information of both similarity and dissimilarity among data points.

An implementation of the MI3 approach may integrate semi-supervised learning algorithms for label propagation. In particular, we designed a graph-based label propagation process based on Zhou et al.'s method [87] as one of the optional algorithmic default labeling components in our implementation of MI3 pipelines. Typical unsupervised methods, such as clustering, may also be used for assisting the label propagation.

Visualization can assist users in performing their tasks in active learning workflows. For example, Höferlin et al. presented an active learning pipeline for training classifiers in the context of video analytics [31]. Their active learning pipeline contains three stages: (i) algorithmic sampling with an option for user sampling with the aid of visual analytics, (ii) manual annotation, (iii) training classifier with an option for manual modification of a classifier with the aid of visual analytics. Bernard et al. proposed an active learning workflow for learning a similarity metric in sports data analysis [4]. It includes three components: (i) visual-interactive learning interface, (ii) similarity modeling by learning weights of features, and (iii) a list-based result visualization that ranks data objects by the learned distance.

Building on the two case studies [4, 31], Bernard et al. outlined a conceptual framework for visual interactive labeling [5], which features six components: (i) preprocessing and feature extraction, (ii) learning model, (iii) result visualization, (iv) candidate suggestion, (v) labeling interface, and (vi) feedback interpretation. This framework advocates the close integration between interactive visualization with active learning to enable concurrent objectives, including data labeling, model training, and knowledge acquisition by the user.

Other conceptual frameworks relevant to this work include:

- The conventional active learning framework summarized by Settles following a survey of the active learning literature [72]. It includes three main stages: (i) algorithmic sampling, (ii) interactive annotation, and (iii) training classifier.
- The human-centered ML framework by Sacha et al. [69], which includes (i) data editing and enrichment, (ii) data preparation, (iii) model selection and building, (iv) exploration and direct manipulation, and (v) execution and evaluation.
- The VIS4ML ontology by Sacha et al. [68], which divides ML workflows to four stages: (i) prepare data, (ii) prepare learning, (iii) model learning, and (iv) evaluate model, and further lists 24 processes in ML that can benefit from interactive visualization.
- The general pathway from data mining to knowledge discovery by Fayyad et al. [22], which includes (i) goal identification, (ii) dataset creation, (iii) cleaning and reduction, (iv) matching KDD goal with a data mining method, (v) exploratory analysis, (vi) data mining, (vii) result interpretation, and (viii) action based on the discovered knowledge.
- Various frameworks for visualization and interaction, e.g., the operator interaction framework by Chi and Riedl [18], the data state reference model for visualization by Chi [17] and interactive visualization workflows by Card et al. [11] and van Wijk [82].

The active learning approach described in this work falls into the conceptual frameworks of Bernard's and Sacha's [5, 68, 69]. It focuses on data reconstruction applications, labeling multiple types of data, learning with sparse data, and interaction minimization. In each active learning pipeline, every decision made by human users is primarily for the objective of reconstructing data as quickly as possible and with as few interactions as possible. The effort for the users to "teach" an ML model by correcting its errors is essentially to enable the model to provide useful assistance to the users as soon as possible. The users make simple decisions to correct the model's errors and make complex decisions as to the right moment to stop "teaching" the model.

Different active learning pipelines are used to detect different types of data objects. It is desirable to have a common approach to develop these pipelines. Having similar operational flows, visual interfaces, and interaction designs can itself reduce the interaction cost. We detail this common technical approach in Sections 3 and 4, and describe three pipelines in the application context of John Snow's cholera map in Section 5. In Section 6, we demonstrate how the cost of interaction is reduced with different algorithmic solutions. In Section 7, we discuss the potential extension of these three pipelines to some 18 pipelines and reuse them for many other reconstruction applications.

3 OVERVIEW OF THE MI3 APPROACH

In this section, we first describe an example case that illustrates the challenges in a class of data reconstruction applications. We then outline a generic approach for supporting such applications.

3.1 Examples of Technical Challenges: John Snow's Cholera Map

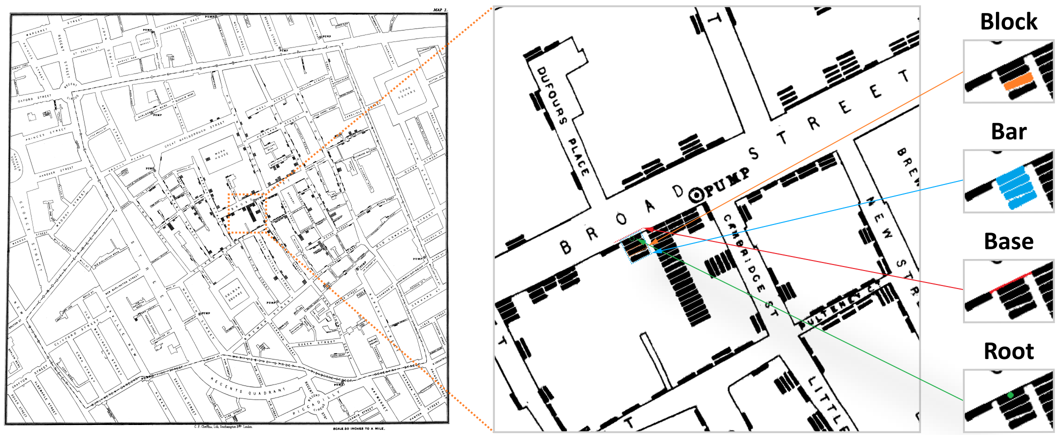


Fig. 1. The cholera map [74] shows the number of fatalities in the 1854 London Cholera Outbreak with a discrete bar chart. The fatalities are distributed at more than 300 locations on the map and visualized as stacked blocks. The right figure illustrates the visual encoding of the cholera map. A bar aggregates the fatalities at a location, where the number of blocks denotes the number of fatalities. A bar typically sits on a base and has a root point that denotes its location information.

The 19th-century witnessed the boom of visualization techniques. Among many well-known historical visualizations created at that time, the cholera map created by John Snow in 1854 [74] provides a telling example for illustrating the challenges in data reconstruction from historical visualization images. As shown in Fig. 1, the map depicts the spatial distribution of fatalities in an area during the 1854 London cholera outbreak.

The cholera map encodes a precious historical dataset, which can be stored as an array of records, each of which is a tuple $(x, y, \#victims)$. The goal of data reconstruction is thus to scan the map, identify the location (x, y) of each bar with at least one block, and count the number of blocks as the value of $\#victims$. As illustrated on the right of Fig. 1, the identification of each location (x, y) requires several object recognition processes for identifying the corresponding bar, the blocks that form the bar, the base that determines the first block, and the root position (i.e., a consistently-defined position in the first block).

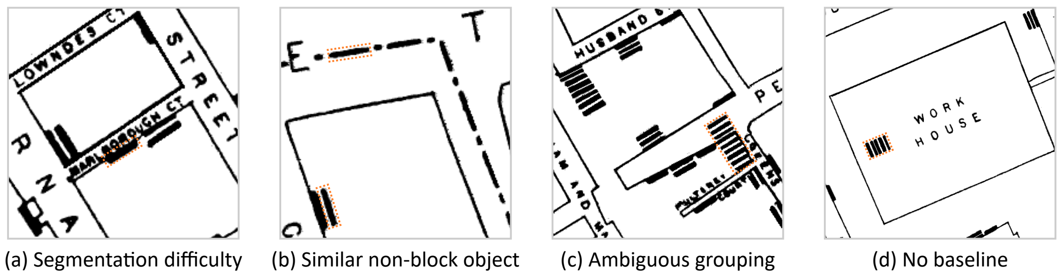


Fig. 2. Some examples of challenging cases in processing the cholera map image [74]: (a) A block conglutinates with background; (b) A segment of dashed line may be confused for a block; (c) The grouping of blocks is ambiguous; (d) Blocks are not distributed close to a baseline.

The cholera map consists of 579 blocks that form 321 bars. Manually acquiring the values for the 321 records will be tedious and time-consuming. Even with an optimistic estimation, one were to spend 10 seconds to navigate to a bar, 10 seconds to count the number of blocks in the bar, 20 seconds to measure the location, and 10 seconds to type in the values into a spreadsheet, it would take some 50 seconds per bar and more than 4 hours to reconstruct the dataset.

Naturally, one would like to automate this process as much as possible. However, data reconstruction from historical visualizations poses several challenges. Many of such visualizations were hand-drawn, often featuring unique visual designs and suffering from poor image quality. Any general-purpose algorithms for detecting blocks, bars, bases, or roots may deliver erroneous results when encountering difficult patterns such as those shown in Fig. 2. In Fig. 2(a), a block conglutinates with the baseline and the letters on the other side. In Fig. 2(b), a segment in a dashed line can easily be confused as a block. In Fig. 2(c), the grouping of blocks into bars is ambiguous, even for humans. At a glance, the highlighted pattern may be interpreted either as a bar with nine blocks or as two bars with four and five blocks respectively. It requires careful observation of other bars in the neighborhood, and some logical reasoning in order for one to conclude that this is a bar of nine blocks. In Fig. 2(d), a bar does not have a baseline because it is not associated with a street location. Hence, the identity of its first block is ambiguous, and so is its root position.

While one may anticipate the potential of using machine learning to train a sequence of models for detecting blocks, grouping blocks into bars, and finding the root of each bar respectively, the unavailability of a sufficient number of training data points poses another critical challenge. Many historical visualizations, including the cholera map, feature unconventional visual designs. Given such a visualization image, the data objects (e.g., blocks, bars, or roots) that may be prelabeled to aid a supervised learning process would constitute a very sparsely sampled and possibly biased training dataset if there were other visualizations drawn with the same visual design. Therefore, prelabeling the 579 blocks, 321 bars, and 321 roots would unlikely cost less than measuring the 321 tuples $(x, y, \#victims)$ manually and typing them into a spreadsheet.

Hence, it is highly desirable to find an approach that is neither totally automated nor totally manual. Ideally, a software system enabling such an approach encodes some basic knowledge about a family of plots (e.g., bar charts, discrete bar charts, pictogram bar charts). It asks a user questions about the difficulties and variations specific to a given visualization (e.g., noise and distortion, and root definition), and dynamically learns from the user's answers and improve its ability to handle similar difficulties and variations. Like an intelligent assistant, it initiates interactions intelligently and learns dynamically. This desire motivates the development of the MI3 approach, where MI3 stands for "Machine-Initiated Intelligent Interaction".

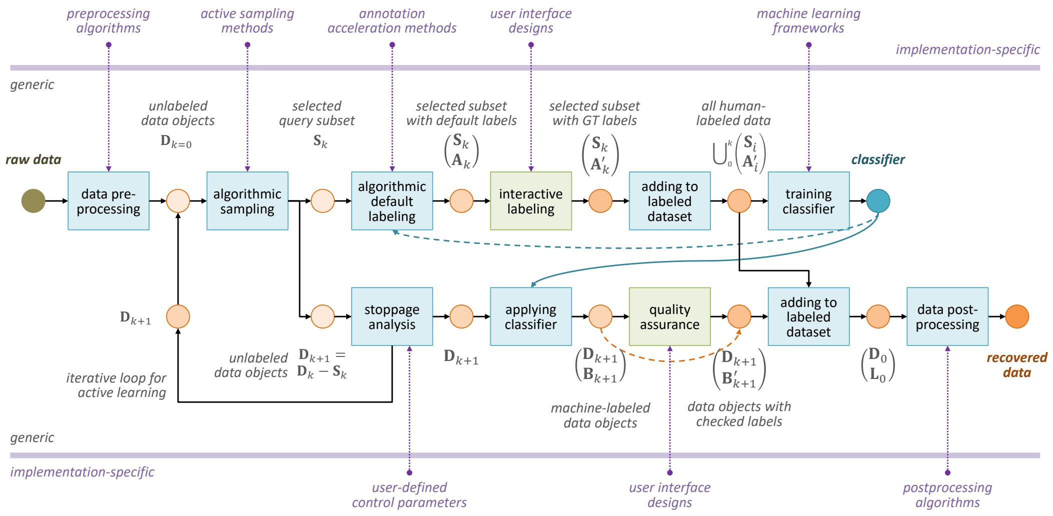

preprocessing algorithms
active sampling methods
annotation acceleration methods
user interface designs
machine learning frameworks
implementation-specific
generic
raw data
data pre-processing
unlabeled data objects $\mathbf{D}_{k=0}$
algorithmic sampling
selected query subset $\mathbf{S}_k$
algorithmic default labeling
selected subset with default labels $\binom{\mathbf{S}_k}{\mathbf{A}_k}$
interactive labeling
selected subset with GT labels $\binom{\mathbf{S}_k}{\mathbf{A}'_k}$
adding to labeled dataset
all human-labeled data $\bigcup_0^k \binom{\mathbf{S}_i}{\mathbf{A}'_i}$
training classifier
classifier
$\mathbf{D}_{k+1}$
iterative loop for active learning
stoppage analysis
unlabeled data objects $\mathbf{D}_{k+1} = \mathbf{D}_k - \mathbf{S}_k$
applying classifier
$\mathbf{D}_{k+1}$
quality assurance
$\binom{\mathbf{D}_{k+1}}{\mathbf{B}_{k+1}}$
machine-labeled data objects
$\binom{\mathbf{D}_{k+1}}{\mathbf{B}'_{k+1}}$
data objects with checked labels
adding to labeled dataset
$\binom{\mathbf{D}_0}{\mathbf{L}_0}$
data post-processing
recovered data
generic
implementation-specific
user-defined control parameters
user interface designs
postprocessing algorithms

Fig. 3. The generic MI3 pipeline. The blue blocks denote algorithmic steps, and the green blocks denote interactive steps. The optional data preprocessing step transforms the raw data from the original problem into a classification task for a set of unlabeled data objects $\mathbf{D}_0$. We consider the $k+1$-th iteration of the active learning as an instance. Here, by "active learning", we refer to its general definition that the system actively seeks decisions from the user to aid the ML process. The algorithmic sampling step selects the data objects $\mathbf{S}_k$ to be labeled. The algorithmic default labeling step assigns default label annotation $\mathbf{A}_k$ for data objects $\mathbf{S}_k$. In the interactive labeling step, the user corrects the mislabeled data objects in the interface and produces ground truth (GT) labels $\mathbf{A}'_k$. The confirmed label annotations $\mathbf{A}'_k$ are then added to the labeled dataset. An interim classifier is trained with the partially labeled dataset, which can be used to compute default labels in the next iteration. The stoppage analysis step checks whether the interactive classification stage should stop. If not, the next session of interactive classification starts. Else, the interim classifier can be applied to label the remaining unlabeled data objects $\mathbf{D}_{k+1}$ with labels $\mathbf{B}_{k+1}$. The user goes through a quality assurance step to verify the labels for these data objects. The data objects $\mathbf{D}_{k+1}$ and verified labels $\mathbf{B}'_{k+1}$ are then added to the labeled dataset and forms the labels $\mathbf{L}_0$ for the whole data object set $\mathbf{D}_0$. An optional postprocessing step transforms the labeled data objects into the desired final output data structure. The MI3 pipeline is generic, and its components can be implemented with application-specific algorithms.

3.2 The Generic MI3 Pipeline for Interactive Classification

The goal of an MI3 pipeline is to perform an interactive classification task. Many detection problems in data reconstruction are inherently classification problems. For example, given a collection of pixel-based objects in the cholera map, to differentiate those blocks that represent fatalities from other shapes is a typical classification problem. Even when a detection problem may not immediately appear as a classification problem, it can normally be decomposed into a classification problem plus some preprocessing and/or postprocessing. For example, the problem of grouping blocks into a bar can be transformed to a classification problem for determining whether two blocks are related (i.e., belong to the same bar), together with a preprocessing step for compiling candidates of block pairs and a postprocessing step for grouping those related blocks. Both the preprocessing and postprocessing steps can be reliably computed using predefined algorithms without any user intervention. By focusing on classification problems, the MI3 approach can be applied to different pipelines and sub-pipelines in data reconstruction workflows.

Fig. 3 depicts the MI3 approach as a generic pipeline. Let $\mathbf{D}_0$ be a set of n unlabeled data objectives at step 0 (i.e., before the iteration starts). The goal of the pipeline is to deliver a labeled dataset $\binom{\mathbf{D}_0}{\mathbf{L}_0}$ after $k+1$ iteration steps such that the amount of human effort in these $k+1$ steps is less than that for labeling all n data objects manually. In this work, we use the number of human-computer interactions to approximate the amount of human effort.

In the pipeline, *data preprocessing* is an optional process, and it is used when the raw data has not yet provided a set of data objects to be labeled. For example, given the cholera map image, one may use a preprocessing step to extract all color connected pixel components (i.e., all connected black pixel groups and all connected white pixel groups) as candidates of blocks. Normally, this preprocessing step is totally automated. The subsequent processes will classify these candidates into correct categories, such as "true blocks" and "false blocks". It is thus important in practice for this preprocessing step to minimize false negatives.

As all data objects in $\mathbf{D}_0$ are unlabeled, the core of the pipeline is an iterative loop based on active learning (according to its general definition). Consider the $k+1$-th iteration as an instance. The data flows through various component processes as follows:

- The process *algorithmic sampling* selects a subset of data objects $\mathbf{S}_k \subseteq \mathbf{D}_k$. While a simple solution can be random sampling, a more sophisticated solution can select data objects that can inform the classifier learning more effectively.
- The process *algorithmic default labeling* assigns a tentative label to each object in $\mathbf{S}_k$. We denote this interim labeled subset as $\binom{\mathbf{S}_k}{\mathbf{A}_k}$. It is helpful for those tentative labels to be as correct as possible. All incorrect labels will have to be corrected by the succeeding process manually, hence incurring more interactions.
- The process *interactive labeling* then initiates an interaction session, asking the user to check the interim labeled subset and make a correction if necessary. This results in a correctly labeled subset $\binom{\mathbf{S}_k}{\mathbf{A}'_k}$. The main criterion for designing an effective user interface is to enable the checking of the tentative labels and correcting any mistakes with minimal human effort. In this work, we focus on the number of interactions as an approximation of human effort.
- In the process *adding to labeled dataset*, the labeled subset $\binom{\mathbf{S}_k}{\mathbf{A}'_k}$ is then combined with all labeled subsets in the previous k iterations, resulting in a larger set of labeled data objects $\bigcup_0^k \binom{\mathbf{S}_i}{\mathbf{A}'_i}$, which can be used to train a classifier while forming part of the classification results that the pipeline is designed to deliver. Here we define the union of a series of set-pairs as the pair of two sets, each resulting from the union of the corresponding series of sets.
- The process *training classifier* is an automated ML process that uses the labeled data objects $\bigcup_0^k \binom{\mathbf{S}_i}{\mathbf{A}'_i}$ to train a classification model. Because this training process is iteratively invoked with bigger and bigger training datasets, the ML models are expected to become better and better. The interim model can be used to classify unlabeled data objects in the processes of *algorithmic default labeling* and *applying classifier.*
- The process *stoppage analysis* makes an algorithmic decision as to the trained model obtained after the $k+1$ iteration steps is good enough for classifying unlabeled dataset $\mathbf{D}_{k+1}$. This decision can be made using the information collected in the previous $k+1$ iterations, e.g., the testing measures in the process *training classifier*, the percentage of the labels that need to be corrected in the process *interactive labeling*, and some statistical measures about the labeled data and unlabeled data. If the trained classifier is equipped with uncertainty analysis, there can be an additional *stoppage analysis* process after the process of *applying classifier.*
- The process *applying classifier* is invoked once the *stoppage analysis* gives the green-light. As $\mathbf{S}_k$ is selected from $\mathbf{D}_k$, the set of remaining data objects yet to be labeled is denoted as $\mathbf{D}_{k+1}$.

Using the trained model, the process labels all the data objects in $\mathbf{D}_{k+1}$, resulting in a labeled dataset $\binom{\mathbf{D}_{k+1}}{\mathbf{B}_{k+1}}$.

- The process of *quality assurance* is necessary for any application that demands a very high quality of data reconstruction. It is an interactive process for a human user to inspect the machine-labeled data objects in $\binom{\mathbf{D}_{k+1}}{\mathbf{B}_{k+1}}$ and correct all errors found. This results in a checked and corrected dataset $\binom{\mathbf{D}_{k+1}}{\mathbf{B}'_{k+1}}$.
- The second process of *adding to labeled dataset* combines $\binom{\mathbf{D}_{k+1}}{\mathbf{B}'_{k+1}}$ and $\bigcup_0^k \binom{\mathbf{S}_i}{\mathbf{A}'_i}$ to form the final set of labeled data objects, which is denoted as $\binom{\mathbf{D}_0}{\mathbf{L}_0}$.

Similar to *data preprocessing*, *data postprocessing* is an optional process for transforming the labeled data objects $\binom{\mathbf{D}_0}{\mathbf{L}_0}$ to those in an intended data structure and format. In some applications, an MI3 pipeline may be followed by another MI3 pipeline. We will see examples of *data preprocessing*, *data postprocessing*, and multiple pipelines in the following sections.

4 INSTANTIATING MI3 PIPELINES IN PRACTICE

In this section, we use the process of designing a software system for reconstructing data from discrete bar charts to showcase how to design an interactive classification process following the generic MI3 pipeline in practice. The cholera map shown in Fig. 1 typifies such a visual representation. The data reconstruction system consists of three pipelines, all of which were designed and implemented following the generic MI3 pipeline. They are pipelines for (i) detecting objects (e.g., detecting blocks in the cholera map), (ii) grouping components (e.g., detecting bars in the cholera map), and (iii) determining key positions (e.g., detecting roots in the cholera map).

4.1 Transforming a Decision Problem to a Classification Problem

To benefit from the generic MI3 approach, one needs to transform various decision problems such as detecting objects, grouping components, and determining key positions to classification problems. As mentioned in Fig. 3, we can use a preprocessing step to generate a list of candidates representing potentially correct decisions. These candidates then become the inputs of a classification pipeline, which categorizes these candidates into different label classes.

For example, when the first MI3 pipeline is activated for detecting blocks in the cholera map, the preprocessing step may construct a list of candidates, each of which is a group of connected pixels. Thus, the function of the pipeline is to label each candidate as a valid "block" representing a victim, or a "non-block". All candidates labeled as "block" are then passed to the next pipeline for grouping these blocks into bars.

When the second MI3 pipeline is activated for grouping the detected blocks into bars, the preprocessing step may generate a list of pairwise relations, each indicating that a pair of blocks may potentially belong to the same bar. These candidate relations are then fed into the interactive classification pipeline for labeling each candidate as "yes" for belonging to the same bar, and "no" otherwise. In this case, a postprocessing step is necessary for transforming the results of the classification problem to the results of the original grouping problem. This can be achieved with a simple algorithm converting detected blocks and confirmed relations to a list of bars, each consisting of a group of blocks linked with the confirmed relations. This postprocessing step can also count the blocks in each bar, resulting in the data value *#victims* for each bar.

When the third MI3 pipeline is activated for determining the root positions of the bars, the preprocessing step may generate a set of potential positions for each bar, such as the center of the bar and that of each block in the bar, the middle point of each edge of the bar and its blocks, and the corners of the bar and its blocks. The classification pipeline then labels the candidate positions as “root” and “non-root”, while ensuring exactly one root per bar. A postprocessing step then converts the local root position relative to the bar to a global position (x, y) relative to the original visualization image. In combination with the data value $\#victims$ for each bar, the system generates a list of tuples in the form of $(x, y, \#victims)$.

If the user requires the actual geographical location (e.g., longitude and latitude, or other geodetic systems) for each tuple, this can be done trivially using an image-specific coordinate transformation as a postprocessing step.

4.2 Variations of MI3 Pipelines

The design and implementation of an MI3 pipeline in practice depend on many factors, such as the technical availability of various algorithmic components, the knowledge and skills of the developers, the cost and time constraint of the software life cycle, and many other application-specific requirements. In general, one can consider the development as an agile software engineering approach, with a gradual introduction of better algorithms, techniques, or user interfaces for different processes in Fig. 3. One may consider some typical variations of the processes in the interactive classification part of MI3 pipelines:

- Variations of *algorithmic sampling* — Perhaps the most naive sampling method, which is denoted as **NS**, is to fetch data objects to be labeled simply according to their order in the input dataset $\mathbf{D}_k$. More commonly used in ML, *random sampling* (**RS**) is considered as the baseline sampling method, which selects data objects for labeling in a stochastic order. The approach of *algorithmic sampling* (**AS**) selects data objects according to a predefined metric that predicts the potential benefit of labeling a data object to the learning process. Such a metric may assess the potential benefit according to the uncertainty of the label [9], the diversity of the samples [84], clustering information [54, 84], and the expected performance of the classifier [28]. In this work, we used two methods, including Lewis and Catlett’s uncertainty-based sampling [44] and Xu et al.’s sampling method according to weighted uncertainty, cluster density, and sample diversity score [84].
- Variations of *algorithmic default labeling* — When data objects are presented in a user interface for dynamic labeling, the system may assign a default label to each data object. If the system can predict some labels correctly, this can reduce the number of interactions that a human user has to perform in labeling the data objects. A most naive approach is *no default* labeling (**ND**). An alternative approach is *random default* labeling (**RD**), which assigns a data object to one of the label classes in a stochastic manner. The approach of *algorithmic default* labeling (**AD**) uses a model to predict the label of a data object. For example, one may use the interim model during an ML process to label a data object. Alternatively, one may use a label propagation algorithm (e.g., self-training [85], co-training [7], graph-based label propagation [87, 89]) to predict the label of a data object according to those data objects that have already been labeled. In this work, we have instantiated our MI3 pipelines with the options of using the interim model (**AD (Interim)**) as well as a graph-based label propagation algorithm (**AD (Graph)**) [87] for default labeling.
- Variations of *interactive labeling* — The approach of active learning represents a small portion of all research papers published on ML. One can naturally consider that the baseline approach in ML is to prelabel all data objects in the training and testing datasets, without any interactive labeling during an ML process. This baseline approach is denoted as *prelabeling* (**PL**). Since the MI3 approach is designed for active learning, the process of interactive labeling (**IL**) is expected to be present in almost all MI3 pipelines. Different designs of the user interface for interactive labeling will impact an MI3 pipeline differently. It is desirable to explore many

design options. In this work, we mainly focused on using the simplest interaction modality of button clicking to minimize the time and cognitive load per interaction.

- Variations of *training classifier* — We consider that the baseline is without any automatic and semi-automatic ML, and all data objects will be labeled manually. We denote this *brute force* approach as **BF**, while denoting the fully automatic and semi-automatic ML approaches as **ML**. In practice, there are many design options for an ML pipeline. The MI3 approach does not impose any restrictions. For example, MI3 pipelines can be instantiated with different ML frameworks for classification, e.g., variations of neural networks, decision trees and random forests, support vector machines, and Bayesian networks. In this work, we have used decision tree and the transductive model learned in label propagation [87] as two options for the process *training classifier*.

In Section 5, we will provide further technical details of the algorithms implemented in a software prototype built based on the MI3 approach. In Section 6, we will compare how different combinations of the above variations may impact the performance of the implemented MI3 pipelines.

5 DATA RECONSTRUCTION FROM CHART IMAGES

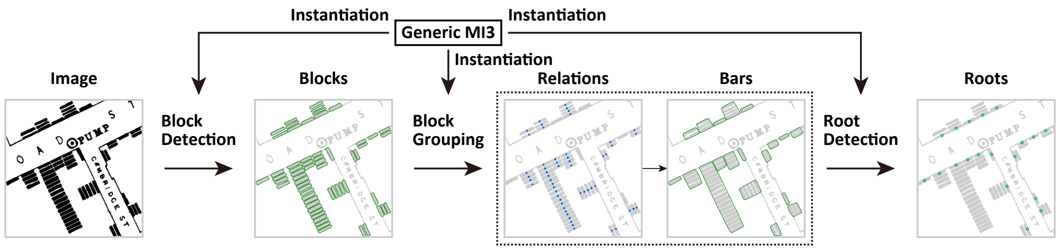


Fig. 4. A data reconstruction procedure for spatial data visualization with three subroutines: block detection, block grouping, and root detection. Each of the three subtasks is an interactive classification task, and is solved with an instantiation of the generic MI3 pipeline. **(1) Block Detection:** Given an input image, we use connected components detection as a preprocessing to convert block detection into a classification task for connected components. **(2) Block Grouping:** With the detected blocks, we build a graph of block relations and classify the existence of relations to determine which pairs of blocks should be grouped in the same bar. The confirmed relations are used to compile the detection of bars. **(3) Root Detection:** For each bar, we generate candidate root points. Therefore, the task of determining the key point is transformed into the classification of candidate roots.

To validate the feasibility of the MI3 approach, and to study and evaluate different design options in realizing MI3 pipelines, we have developed a prototype software system for reconstructing data from a family of visual representations, which are composed of a collection of spatially distributed similar data objects. Each data object depicts a data tuple $(x, y, \#value)$. The spatial location (x, y) may be geographically meaningful and related to a background map or image. The variable $\#value$ may be encoded as the size of the data object, the number of components inside the data object, or other attributes. The cholera map shown in Fig. 1 exemplifies such a visual representation. Fig. 4 shows the procedure of data reconstruction from the discrete bar chart. We decompose the data reconstruction task into three subtasks: block detection, block grouping, and root detection. In the following subsections, we first detail the preprocessing steps that transform the three subtasks into classification problems. We then describe the active learning components shared by the MI3 pipelines for the three subtasks.

5.1 Block Detection Preprocessing

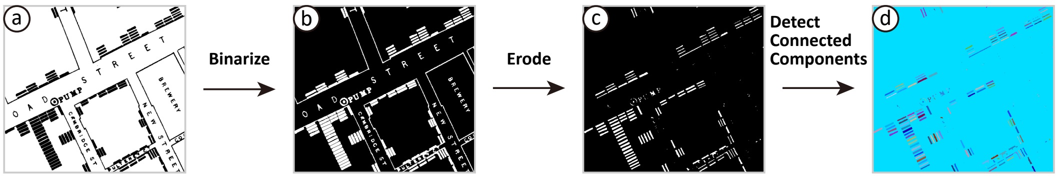


Fig. 5. Candidate block detection process on part of the cholera map [74]: (a) Input part of the cholera map; (b) Binarize the image to reduce color variations; (c) Erode the image to reduce conglutination; (d) Detect connected components.

Following the generic MI3 pipeline, we need to transform the block detection problem into a classification problem. For this transformation, we need to first generate candidate data objects (i.e., in this case, connected components) to be classified, and then compute the feature representation of each data object to facilitate the training of the interim classifier.

Candidate Data Object Detection: We design a preprocessing algorithm, as illustrated in Fig. 5. With the observation that blocks differ from the background in color, we let connected components serve as candidate blocks. The candidate block detection procedure is as follows:

(1) Binarize the image with Otsu thresholding to eliminate minor color variations (Fig. 5(b)).
(2) Erode the binary image with 3×3 cross kernel for two iterations to reduce the conglutination of elements (Fig. 5(c)).
(3) Detect connected components (Fig. 5(d)).
(4) Dilate each connected component separately using the cross kernel for two iterations to compensate the size shrink during erosion.

Feature Computation: For each candidate block, we compute two sets of features: 13 appearance features and 14 neighborhood features, which sums to 27 features. Appearance features capture the visual appearance of the candidate block. Neighborhood features capture the image features of the candidate block's neighborhood.

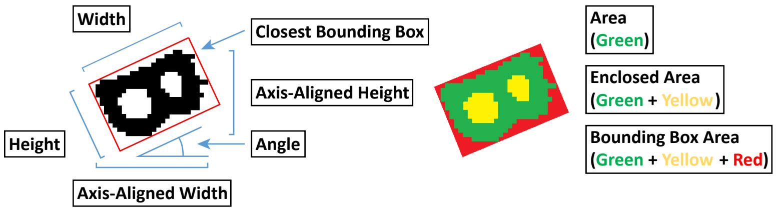


Fig. 6. Measurements related to appearance features of candidate blocks. The left shows the size features and the angle. The right shows area definitions involved in the computation of shape features.

We compute size features, shape features, and color features as appearance features to capture the appearance of a candidate block. Size features include *width*, *height*, *axis-aligned width*, *axis-aligned height*, and *area*, which are all measured in pixels. Shape features include $solidity = \frac{area}{enclosed\ area}$, $convexity = \frac{area}{convex\ hull\ area}$, $extent = \frac{area}{bounding\ box\ area}$, $aspect\ ratio = \frac{width}{height}$, and *angle*. Fig. 6 shows visual attributes used as size features and attributes used for computing shape features. Color features include *r*, *g*, and *b* computed by averaging the color of all the pixels. In total, 13 appearance features are computed.

The 14 neighborhood features are mainly based on the distribution of pixel colors in the candidate block's neighborhood. Multiple neighborhood definitions are considered in the computation. We describe the details of neighborhood features in Appendix A.

5.2 Block Grouping Preprocessing

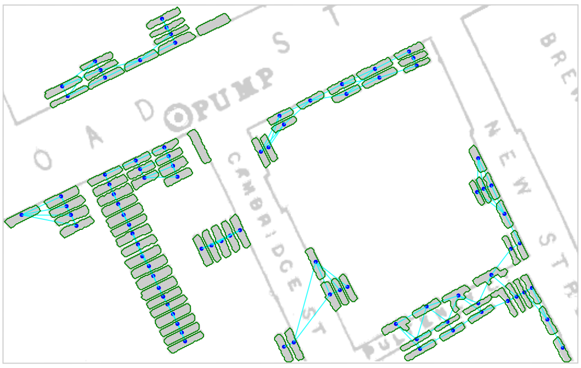


Fig. 7. Candidate neighboring relation detection result on part of the cholera map [74]. The blocks are denoted with green contours. The blue dots show the center of each block. The detected candidate neighboring relations are shown as cyan lines linking the centers of the two blocks involved.

To aggregate the number of victims according to location, we need to group neighboring blocks into bars. We reduce the grouping problem to determining which pairs of blocks are neighbors. If we regard each block as a node and each possible neighboring relation as an edge, the grouping problem is transformed into graph edge classification, which fits the MI3 pipeline. Fig. 7 highlights the detected blocks and candidate neighboring relations.

Candidate Data Object Detection: Each pair of blocks is potentially a pair of neighboring blocks. Given n blocks, there are in total $\frac{1}{2}(n^2 - n)$ candidate relations. In practice, the quadratically exploding number of data objects pose a computation challenge for algorithms. Therefore, we design two heuristic filtering rules as follows:

(1) There is no relation between a pair of blocks when they are not mutually the top four closest to the other block among all the blocks.
(2) There is no relation between a pair of blocks when the distance between the two blocks is larger than the length of the longest axis of all blocks.

Feature Computation: For each candidate neighboring relation, we compute *position distance*, *color distance*, and *overall distance* as the features. For a pair of blocks, the *position distance* is defined as the Euclidean distance between the position of the two blocks (x_1, y_1), (x_2, y_2) in pixels. For a pair of blocks, the *color distance* is defined as the Euclidean distance between the two blocks in the (r, g, b) space. The *overall distance* is defined as the root mean square of the *position distance* and *color distance*.

5.3 Root Detection Preprocessing

To reconstruct the position information of a data object in the spatial visualization, we need to detect the data object's representative point. To align this problem to the MI3 pipeline, we need to generate a set of candidate roots to be classified, where a "root" refers to the key point in the data object that corresponds to its position.

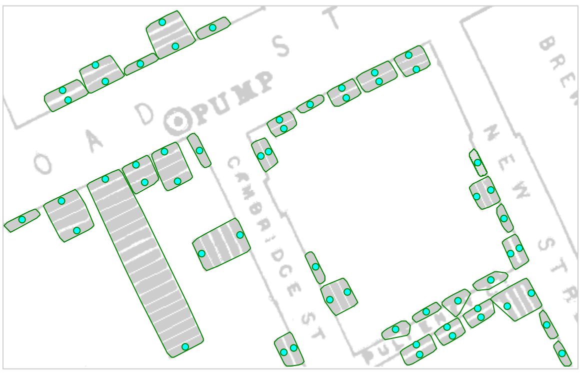


Fig. 8. Candidate root detection result on part of the cholera map [74]. The bars are denoted with green contours. Each candidate root is denoted with a cyan dot.

Candidate Data Object Detection: We simplify the root detection problem as finding the center of a block in a bar that represents the position. The root of a bar is an extreme point of the bar that is typically closest to boundary lines in the image. Therefore, we take the center of the topmost and the bottommost blocks of the bar to be candidate roots. Fig. 8 shows the detected candidate roots on part of the cholera map.

Feature Computation: For each candidate root, we compute in total 9 neighborhood features. We observe that roots are typically close to boundary lines in the image, and that roots of multiple bars may be aligned. Therefore, we reuse some of the neighborhood features of block as root features, including *horizontal neighbor distance*, *surrounding neighbor number*, *surrounding foreground rate*, *horizontal foreground rate*, *host area* and *host solidity*. Besides, we compute a boolean feature *single* to denote whether the block containing the root is the only block in a bar. When a root is *single*, it is always a true detection. We also compute two additional neighbor distribution features: *distance to alignment line* and *distance to skeleton*. The two features are described in detail in Appendix B.

5.4 Interactive Classification and Active Learning

In the previous subsections, we illustrate how to transform subtasks of data reconstruction into interactive classification problems that MI3 concerns. In the following, we introduce how the two major algorithmic components of MI3, algorithmic sampling and algorithmic default labeling, are instantiated in our implementation. We also introduce our prototype implementation of an MI3-based data reconstruction system, which shows how the major interactive component, interactive labeling, is instantiated. The implementations of algorithmic sampling, algorithmic default labeling, and interactive labeling are reused for all the three subtasks.

5.4.1 Algorithmic Sampling (AS). The *algorithmic sampling* process selects a set of data objects $S_k = (x_{k_1}, ..., x_{k_l})$ to be labeled interactively, where l is the number of data objects shown in the interface at a time. The selected data objects are to be given default labels (see Section 5.4.2), and the data objects and their default labels are to be presented to the user for label correction (see Section 5.4.3). The input to *algorithmic sampling* is unlabeled data objects $D_k = (x_1, x_2, ..., x_n)$ as shown in Fig. 3. We instantiate *algorithmic sampling* with two optional sampling methods: *entropy-based sampling* and *density, diversity, and entropy-based sampling* (abbreviated as **AS (ES)** and **AS (DDES)** respectively) based on two sampling methods in the literature [44, 84].

With *entropy-based sampling*, the metric of sample priority is defined as the uncertainty of the label [44]. The higher the label's uncertainty, the more the label information can inform the classifier learning process, and thus resulting in less interaction cost. The uncertainty of an unlabeled data object x_i's label y_i is quantified by entropy as $H(y_i|x_i) = -\sum_j p(y_i = j|x_i) log(p(y_i = j|x_i))$. When l instances are to be presented to the user, the top l unlabeled instances that maximize the entropy are sampled. In reality, we do not have access to the true posterior probability $p(y|x)$ used in entropy calculation, and therefore it has to be approximated. When the interim model trained in the *training classifier* stage is a probability-based classifier, we directly use the posterior probability estimated by the classifier as the approximate true posterior probability. When the interim model is not a probability-based classifier, e.g., decision tree, we use Zhou et al.'s algorithm [87] to train an accompanying probability-based classifier that serves as a posterior probability estimator.

Entropy-based sampling only aims to maximize the uncertainty of the samples, while *density, diversity, and entropy-based sampling* sets two additional goals for optimization [84]. It requires that in each iteration, the sampled instances should be distant from each other (diverse), and the sampled instances should be representative in the feature space (dense). To this aim, the scoring function of each data object's priority is defined as $score(x_i) = (1 - \alpha - \beta)H(y_i|x_i) + \alpha \sum_{j=1}^{n} \frac{1}{1+d(x_i,x_j)} + \beta \sum_{j \in S_k} (1 - \frac{1}{1+d(x_i,x_j)})$. It maximizes a weighted sum of label entropy, average inverse distance to other unlabeled data objects, and distance to the data objects already sampled in this batch. The data object with the highest score is added to the sampled set S_k, and sampling is conducted iteratively until $|S_k| = l$. In our implementation, we set $\alpha = \beta = \frac{1}{3}$, and the distance function d to measure Euclidean distance.

5.4.2 Algorithmic Default labeling (AD). Through the *algorithmic default labeling* process, default labels are assigned to sampled data objects, which saves the user's effort for labeling. We instantiate *algorithmic default labeling* with two implementations, including *interim-model-based default* (abbreviated as **AD (Interim)**) and *graph-based default* (abbreviated as **AD (Graph)**).

With *interim-model-based default*, an interim classification model is trained and updated each time a batch of newly labeled data points are added to the labeled set. The interim model predicts the default labels for the instances to be presented to the user in the current iteration.

The idea of *graph-based default* is that data points close to each other in the feature space tend to have the same label, and therefore labeled data objects can propagate their labels to unlabeled data objects. The closer a pair of instances are, the more likely the label can propagate between them. We adapt Zhou et al.'s label propagation algorithm [87] for our implementation of the *graph-based default* method, which is initialized with Algorithm 1 and uses Algorithm 2 for incremental update in the iterative labeling sessions.

Algorithm 1 Transition Matrix Computation

```
Require: instances $X = \{x_1, ..., x_n\}$
Ensure: transition matrix $K_{n \times n}$
1: $W \leftarrow [I(i \neq j) exp(-\frac{||x_i - x_j||^2}{2\sigma^2})]_{n \times n}$
2: $D \leftarrow [I(i = j) \sum_{k=1}^{n} W_{i,k}]_{n \times n}$
3: $S \leftarrow D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$
4: $K \leftarrow (I - \alpha S)^{-1}$
5: return $K$
```

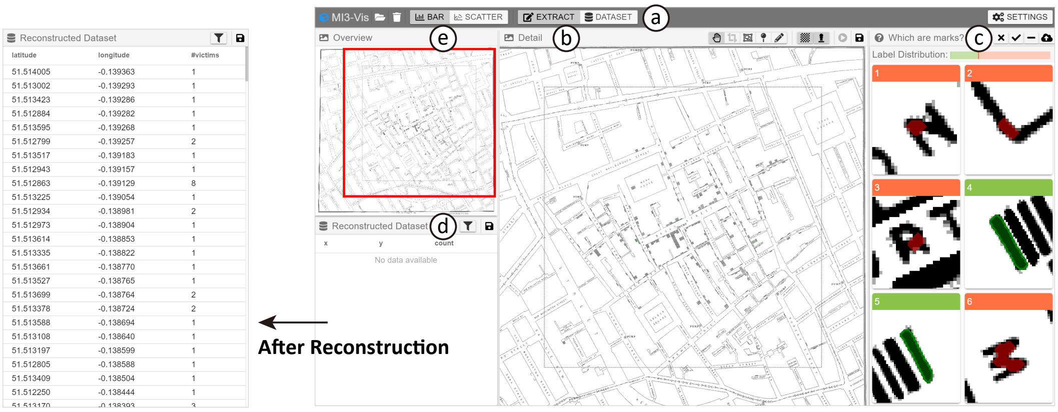
Quantitatively, matrix S in Algorithm 1 denotes the probability of propagation. $S_{i,j}$ is the probability that the label of data object i propagates to data object j in one propagation iteration. This probability is based on the normalized distance between data objects i and j defined as $S_{i,j} = \frac{W_{i,j}}{D_{i,i}}$. $W_{i,j} = exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$ is the unnormalized distance between data objects i and j (where $i \neq j$).

$D_{i,i} = \sum_{k=1}^{n} W_{i,k}$ is the sum of unnormalized distance between data object i and all the other data objects. n is the total number of data objects, no matter labeled or not. The propagation process is run for multiple iterations until convergence. Transition matrix K denotes the transformation that transforms the initial label distribution to the label distribution at the convergent state.

Algorithm 2 Incremental Label Propagation

```
Require: transition matrix $K_{n \times n}$, labels of newly labeled instances $y_{q_1}, ..., y_{q_l}$, label distribution in the last
    iteration $F^t_{n \times c}$ (initialized as $F^0 = 0_{n \times c}$ in the first iteration)
Ensure: updated label distribution $F^{t+1}_{n \times c}$
1: for $q \in \{q_1, ..., q_l\}$ do
2:     $\Delta F \leftarrow [I(j = y_q)K_{i,q}]_{n \times c}$
3:     $F^{t+1} \leftarrow F^t + \Delta F$
4: end for
5: return $F^{t+1}$
```

Each time the user confirms or corrects a batch of l data objects, the algorithm propagates the labels using the transition matrix K, as shown in Algorithm 2 to update the default labels incrementally. In the label distribution matrix $F^t_{n \times c}$, element $F^t_{i,j}$ denotes the estimated probability of whether data object i belongs to class j at the start of the t-th iteration of interactive labeling. c is the total number of classes in the classification. Using the updated label distribution F^{t+1}, the algorithm computes the default labels by maximal likelihood.



latitude	longitude	#victims
51.514005	-0.139363	1
51.513002	-0.139293	1
51.513423	-0.139286	1
51.512884	-0.139282	1
51.513595	-0.139268	1
51.512799	-0.139257	2
51.513517	-0.139183	1
51.512943	-0.139157	1
51.512863	-0.139129	8
51.513225	-0.139054	1
51.512934	-0.138981	2
51.512973	-0.138904	1
51.513614	-0.138853	1
51.513335	-0.138822	1
51.513661	-0.138770	1
51.513527	-0.138765	1
51.513699	-0.138764	2
51.513378	-0.138724	2
51.513588	-0.138694	1
51.513108	-0.138640	1
51.513197	-0.138599	1
51.512805	-0.138588	1
51.513409	-0.138504	1
51.512250	-0.138444	1
51.513170	-0.138393	3

Fig. 9. The prototype user interface for interactive classification and active learning. The interface consists of five parts: (a) Control Toolbar: the user can upload the image, select the chart type, switch interface layout, and change the algorithm parameters; (b) Image View: the Image View shows the input image and the re-visualized reconstructed dataset, and the user can navigate by pan and zoom, clip the image, set the scale of the image, manually create visual objects if needed, and conduct spatial filtering of data objects; (c) Annotation Panel: the user can annotate or correct the labels for data objects; (d) Dataset View: the Dataset View shows the reconstructed dataset as a table after the reconstruction process is finished; (e) Image Overview: the Image Overview highlights the currently zoomed region shown in the Image View.

5.4.3 *The Interactive labeling Interface.*

In the interface as shown in Fig. 9, the user first uploads a visualization image and specifies the type of chart that determines the preprocessing method in the Control Toolbar (Fig. 9(a)). Then, the user clicks the start button on the header of the

Image View (Fig. 9(b)) to start the reconstruction. If needed, the user can switch between different implementations of the MI3 algorithmic components with a popup menu for settings.

After the image is uploaded, the user can click the start button to start the data reconstruction. If needed, the user may clip the image to focus the algorithm on a subpart. Once the reconstruction starts, the system goes through the preprocessing, algorithmic sampling, and default labeling. The sampled objects in each iteration are displayed in the Annotation Panel (Fig. 9(c)). After the sampling, the Image View automatically zooms to the area where the sampled objects are located. The currently zoomed area is highlighted in the Image Overview (Fig. 9(e)). If needed, the user can pan and zoom the image in the Image View.

In the Annotation Panel, the user can click on the thumbnail image to flip the label of a data object or interchangeably press the corresponding number key. The user can also click the object in the Image View to change the label. When the label of the object is hard to determine from the thumbnail image, and the user's mouse has been hovering on the thumbnail for more than 500ms, the system automatically zooms into a small area containing the corresponding object to help the user view the object more clearly.

Once the sampled data objects are checked, the user can click the confirm button in the Annotation Panel or press the "enter" key to add the data points to the labeled set. The integrated interactive labeling and active learning process for one MI3 pipeline runs iteratively until the termination criterion is met. After that, the process for the next MI3 pipeline starts.

After all the MI3 pipelines for data reconstruction are executed, the Dataset View (Fig. 9(d)) shows the reconstructed dataset. The user can export the reconstructed dataset in JSON format. The reconstructed dataset is also visualized and superimposed on the input visualization image. The reconstructed visualization can be exported in SVG format.

6 EVALUATION

In the previous section, we introduce multiple optional implementations of MI3 components that generate multiple instantiations of the MI3 pipeline. In this section, we evaluate the performance of different instantiations. We consider six major variations of solutions to the classification task:

- **BF** — The **brute force** approach where the user needs to use pen-and-ruler to measure and record all the data points without the system's intelligent support.
- **ML + x%-PL** — A conventional **ML-based** system where the user **prelabels x%** of the training set. Then, a classification model is trained with the training set, and assigns default labels for all the remaining data points.
- **RS + RD + k-DL** — An instantiation of the MI3 pipeline that exploits **random sampling**, and **random default labeling**. In each active learning session, **k** data points are randomly sampled, randomly assigned a default label, and presented to the user. This provides a reference benchmark to evaluate the algorithmic sampling and default labeling techniques.
- **AS + RD + k-DL** — Similar to the above, except that **algorithmic sampling** is used in place of random sampling.
- **RS + AD + k-DL** — An instantiation of the MI3 pipeline that exploits **random sampling**, and an **algorithmic default labeling** strategy. In each active learning session, **k** data objects are randomly sampled and then default labeled according to the strategy. The number of interactions depends on the AD technique's error rate and the number k of data objects presented in each iteration.
- **AS + AD + k-DL** — Similar to the above, except that **algorithmic sampling** is used in place of random sampling.

Although it is useful to run user studies, the numerous system design options require a large number of subjects and trials in the study, which makes it hard to carry out in practice. Simulation-based approaches are more feasible in such scenarios. Typical simulation-based evaluation methods include Card et al.'s keystroke-level model [12] and the later thread of work establishing the GOMS method [13, 35, 39]. In this section, to evaluate different options of instantiating the MI3 pipeline, we adopt Zhang et al.'s simulation-based evaluation method [86]. Precisely, the number of interactions needed to finish the data reconstruction task serves as the evaluation metric and is estimated by the simulation. The simulation-based evaluation method makes it possible to gather a large number of repeated trials, and the result does not suffer from the variance of human subjects.

6.1 Dataset

We use John Snow's cholera map [74] as the dataset for evaluation. In the cholera map, there are 579 rectangular blocks that form 321 bars. As illustrated in the previous sections, to reconstruct data from this image, the user needs to carry out three classification tasks, i.e., classify candidate blocks, relations, and roots as true or false detections. For the block detection problem, there are 4416 candidate blocks to be classified detected by the preprocessing algorithm based on connected component detection. Among the 4416 candidates, 533 are positive (true detections), and 3883 are negative (false detections). Note that there are 46 blocks missed by the preprocessing algorithm. Therefore, for this classification task, the user needs to make additional effort to annotate these 46 blocks. For the neighboring relation classification problem, there are 868 candidate relations to be labeled where 258 are positive, and 610 are negative. For the root classification problem, there are 450 candidate roots to be labeled where 321 are positive, and 129 are negative.

6.2 Experiment Design

Compared Solutions: In the simulation, we compare the aforementioned six optional solutions to the classification tasks: (i) **BF**, (ii) **ML + 20%-PL + NeD + 6-DL**, (iii) **RS + RD + 6-DL**, (iv) **AS + RD + 6-DL**, (v) **RS + AD + 6-DL**, and (vi) **AS + AD + 6-DL**. Note that in the block detection dataset, most of the data objects are false detections. For the **ML + x%-PL** solution using conventional ML-based system with $x\%$ data points prelabeled, we can enhance it by always prelabeling data points to have negative default (NeD). Therefore, in the experiment, we substitute the conventional ML solution **ML + x%-PL** with **ML + x%-PL + NeD + k-DL**, which is a more competitive benchmark. We set the prelabel rate $x\%$ to be 20%. For all the solutions, we fix the number of data objects k presented in each active learning session to be 6. For conciseness of the result, we fix entropy-based sampling to be the instantiation of active sampling (AS (ES)), and interim decision tree model to be the model for algorithmic default labeling (AD (Interim)).

Evaluation Metric: In the experiment, for each of the three MI3 pipelines, we measure the interaction cost for each optional solution to achieve 100% accuracy in classification. For example, in the block detection problem, there are 579 true blocks. The preprocessing pipeline detects 4416 data points to be classified, where 533 are true detections. For the **BF** solution using pen-and-ruler or its digital equivalent, it would take a minimal 579 interactions to measure and record all the data objects. For the conventional ML solution with **ML + 20%-PL + NeD + 6-DL**, it would require $883 \approx 4416 \times 20\%$ data points to be labeled in the prelabeling session. Approximately 12% of the data points are true detections. Therefore, the negative default strategy (NeD) would produce incorrect default labels for 12% of the 883 data points. It needs $108 \approx 883 \times 12\%$ interactions for correction of incorrect default labels, and $148 \approx 883/6$ interactions for clicking the confirm button to register the corrections. An additional 46 interactions are needed to correct missing blocks in the preprocessing. Our initial testing result shows that with 883 data points labeled, the model achieved 98.837% accuracy for the remaining $3533 = 4416 - 883$ data points. Therefore, $41 \approx (1 - 98.837\%) \times 3533$

interactions are needed to correct the misclassifications. In total, the number of interactions to achieve 100% accuracy for **ML + 20%-PL + NeD + 6-DL** is 343 = 108 + 148 + 46 + 41.

We can estimate the interaction cost for other optional solutions similarly. Note that for the four instantiations of MI3, the interaction cost is dependent on the number of active learning sessions conducted. With more sessions, more interactions are needed for default label correction. Meanwhile, the more data points labeled, the higher the accuracy of the model, which reduces the interaction cost to quality assure and correct the labels for the remaining data points. To show these dynamics, the overall interaction cost can be represented as a function of the number of interactions in active learning sessions. Note that the number of interactions in active learning sessions excludes the interaction cost for the final quality assurance and correction of the remaining data points' labels. For the block detection problem, we run 75 repeated measures and average the results. For the neighboring relation detection and root detection problems, we run 150 repeated measures and average the results.

6.3 Results and Analysis

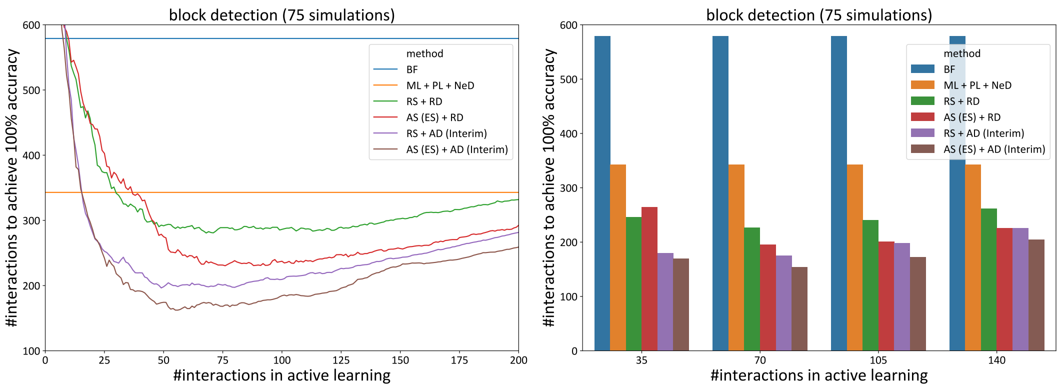


Fig. 10. Comparison of the 6 optional solutions to the block detection problem. The curves on the left show the change of interaction cost to achieve 100% accuracy with regard to the interaction cost in active learning. The bar chart on the right compares the performance of the optional solutions at 35, 70, 105, and 140 active learning interactions. The correspondence between solutions and colored curves and bars is: blue for the brute force approach (**BF**); orange for the conventional ML-based approach with 20% data objects prelabeled and negative default labeling (**ML** + **20%-PL** + **NeD** + **6-DL**); green for random sampling with random default labeling (**RS** + **RD** + **6-DL**); red for algorithmic sampling with random default labeling (**AS** + **RD** + **6-DL**); purple for random sampling with algorithmic default labeling (**RS** + **AD** + **6-DL**); brown for algorithmic sampling with algorithmic default labeling (**AS** + **AD** + **6-DL**). All the solutions except **BF** show 6 data objects in each labeling session, and therefore, the "**6-DL**" is omitted in the legend.

Fig. 10 shows the experiment results in block detection with the six optional solutions. In the following, we discuss patterns observed in the experiment result. For conciseness, we refer to the total interaction cost to finish the classification task (i.e., achieve 100% accuracy) to be the "overall cost", the interaction cost in the iterative active learning sessions to correct the data objects with incorrect default labels to be the "active learning cost", and the interaction cost in the final quality assurance session to correct mislabeled data objects to be "quality assurance cost".

1. Performance of MI3 pipeline instances follows a U-shaped curve. The overall costs of **RS + RD**, **AS + RD**, **RS + AD**, and **AS + AD** follow the same trend of first decrease and then increase with the active learning cost. Such a pattern emerges because the overall cost is comprised of the active learning cost and quality assurance cost. The quality assurance cost typically experiences a rapid decrease during the first few active learning sessions, as the accuracy of a classifier typically improves faster with the increase of training data at the beginning. With more interactions spent on active learning, the rate that quality assurance cost decreases gets slower as the interim classification model's performance reaches a plateau. At some point, the increased contribution of active learning cost to the overall cost would overcome the decrease of the quality assurance cost, making the overall cost increase again.

Because of the U-shaped pattern of the overall cost, for each optional solution, there is an optimal point of the total number of interactions for active learning that minimizes the overall cost. For example, for **RS + RD**, the optimal point is when 68 interactions are spent on active learning. The overall cost at the optimal point depicts the performance of the solution when the iterative labeling process terminates at a suitable time point, as determined by the termination criterion. Assuming that the active learning sessions terminate at suitable time points, we compare different solutions mainly by the lowest point of the curve in the following three observations.

2. MI3 pipeline instances outperform the brute force approach and the conventional machine learning pipeline. The optimal overall costs of **RS + RD**, **AS + RD**, **RS + AD**, and **AS + AD** are smaller than the overall costs of **BF** and **ML + PL + NeD**. For the block detection task, it takes **BF** 579 interactions to record all the blocks. It takes **ML + PL + NeD** 343 interactions to detect all the blocks with 100% accuracy. By comparison, the four MI3 instances have optimal overall costs between $[150, 300]$ interactions.

3. Algorithmic sampling (AS) reduces interaction cost. For MI3 instances with the same default labeling strategy, the instance using an algorithmic sampling strategy outperforms the instance using random sampling. Specifically, the overall cost of **RS + RD** reaches its minimal, 280 interactions, with 68 active learning interactions and then gradually increases. When the sampling method is fixed as **RD**, after incorporating algorithmic sampling, **RS + RD** transforms to **AS + RD**. The optimal overall cost reduces to 230 interactions, which is achieved at 76 active learning interactions. Similar differences are observed between **RS + AD** and **AS + AD**. We interpret the smaller overall interaction cost as algorithmic sampling manages to continuously explore informative samples for the interim model to learn, which sustainably decreases the number of misclassifications and reduces the interaction cost to correct.

4. Algorithmic default labeling (AD) reduces interaction cost. For MI3 instances with the same sampling strategy, the instance using an algorithmic default labeling strategy outperforms the instance using random default labeling. With algorithmic default labeling, a lower overall cost can be achieved with the same number of interactions compared with random default labeling. The overall cost of **RS + RD** at the optimal point is 280 interactions, and is larger than that of **RS + AD**, which is 196 interactions. The reason is that algorithmic default labeling is typically more accurate than random default. Therefore, algorithmic default labeling enables more data points to be labeled with the same active learning cost. Thus, the quality assurance cost of algorithmic default labeling is less than that of random default labeling when their active learning costs are the same.

In summary, instantiations of the MI3 pipeline outperform the brute force approach (**BF**) and the conventional machine learning pipeline (**ML + 20%-PL + NeD + 6-DL**). Moreover, among different instantiations of the MI3 pipeline, we find that for the sampling component, pipelines using algorithmic sampling (**AS**) perform better than those using random sampling (**RS**). For the default labeling component, pipelines using algorithmic default labeling (**AD**) are better than those using random default labeling (**RD**). Among the implemented MI3 instances, the one with the best performance, **AS + AD**, requires a minimal 162 interactions to accomplish the classification task with 100% accuracy, which is less than one-third the cost of the brute force approach, and less than half the cost of the conventional machine learning pipeline.

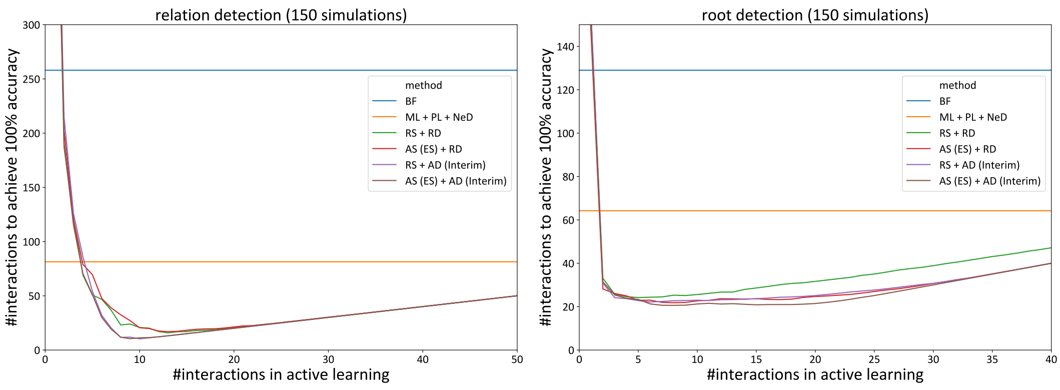


Fig. 11. Comparison of the 6 optional solutions in neighboring relation detection and root detection. The curves show the change of overall interaction costs with regard to the interaction cost in active learning.

Aside from the block detection problem, we also evaluate the six pipelines on neighboring relation detection (for block grouping) and root detection. The experiment results are shown in Fig. 11. We observe that similar to the result of block detection, MI3 pipeline instances outperform the brute force approach and the conventional machine learning pipeline. However, for these two subtasks, algorithmic sampling (**AS**) and algorithmic default labeling (**AD**) bring insignificant benefits over random sampling (**RS**) and random default labeling (**RD**). We interpret it as the result that the two subtasks are much simpler than the block detection problem. Both relation detection and root detection can be accomplished with less than 30 interactions.

7 DISCUSSION: REUSABILITY OF MI3 PIPELINES

Despite that the above MI3 pipelines based on active learning can be more efficient and effective than the conventional machine learning pipeline, there is a development cost associated with each pipeline. Unless each pipeline can be reused for a sufficient number of historical visualization images, the development cost can still be significantly higher than measuring every data point manually and collecting the measured data using a spreadsheet.

We surveyed some ten archive sources, such as Library of Congress [46], United States Census Records [81], David Rumsey Map Collection [66], Old Maps Online [40], Milestones Project [24], and the Historical Atlas of China [76]. There are estimated some 500,000 historical visualizations in various collections around the world. While most of them are geographical maps, there are also thousands of statistical graphics, thematic maps, and illustrations. For the MI3 approach to be developed into a software system that can be applied to most of these visualizations, the number of active learning pipelines required must be significantly smaller than 500,000. If the number of pipelines is moderate, not only is it more cost-beneficial to develop the software system but also likely it is easy and quick for a user to configure a multi-pipeline workflow for a given individual visualization, without the need for any programming effort. In this section, we discuss the feasibility of such a software system by examining the reusability of MI3 pipelines.

Fig. 12 shows nine examples of historical visualizations. With a quick observation, one can easily see that our three pipelines can be reused for several examples. For instance, the block detection pipeline can possibly be used to detect bars in (b) and (c), circles and wedges in (d) and (e), nodes and edges in (f), and dots in (h). The block grouping pipeline can possibly be used to group the bars in a bar-chart glyph in (c) and the wedges in a pie-chart glyph in (e). The root detection pipeline can possibly be used to detect the top-middle location of each bar in (b) and (c), the center-bottom

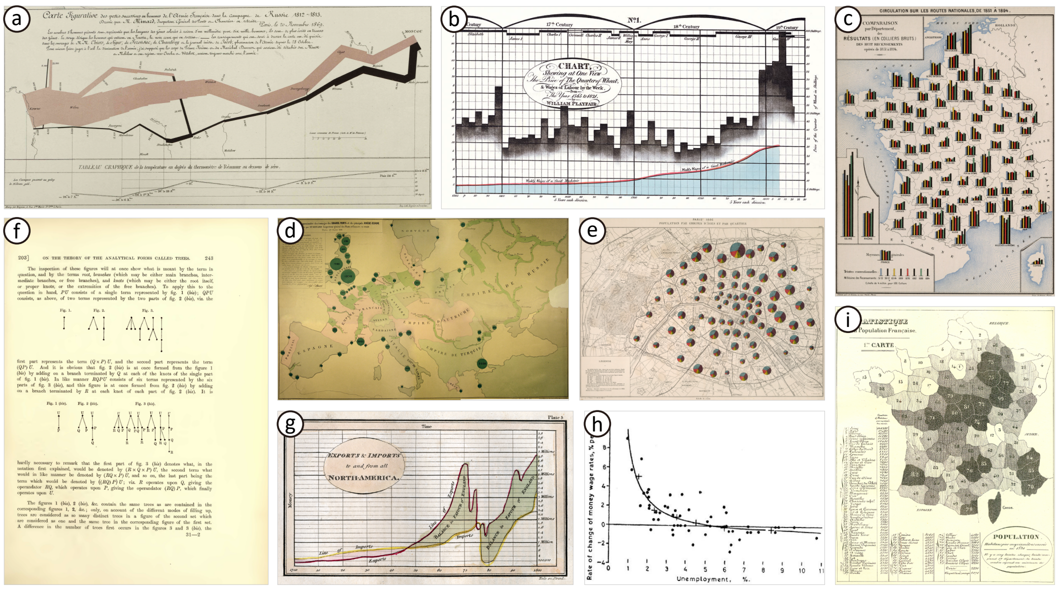


Fig. 12. Examples of historical visualizations: (a) Charles Minard's movement flow map, 1869 [53]; (b) William Playfair's bar chart, 1822 [61]; (c) Émile Cheysson's thematic map with bar-chart glyphs, 1906 [16]; (d) Charles Minard's bubble map, 1859 [52]; (e) Jacques Bertillon's thematic map with pie-chart glyphs, 1886 [6]; (f) Arthur Cayley's node-link diagram, 1890 [14]; (g) William Playfair's line plot, 1784 [59]; (h) William Phillips's scatter plot, 1958 [57]; (i) Adolphe d'Angeville's choropleth map, 1836 [20].

position of each bar-chart glyph in (c), the circular center of each bubble in (d) and each pie-chart glyph in (e), and the center of each node in (f) and each dot in (h).

While each pipeline exhibits reasonable reusability, more pipelines will be needed to transform the MI3 approach to a practical software system. After analyzing over 150 historical visualization images in detail, we identified the following list of 18 pipelines that can be used in combination to support the data reconstruction from numerous historical visualization images.

- **Visual Object Detection**
  - a_1. *Component-based object classification* — given a collection of imagery components, determine if a component is a visual object of a certain type (e.g., blocks in the cholera map, dots in a scatter plot, or nodes in a graph), *while learning a model to do this semi-automatically.* The italicized clause applies to all the following active learning pipelines and is omitted.
  - a_2. *Seed-based object construction* — given a seed on an object, determine all the pixels that belong to the same object (e.g., searching for a region in a map).
  - a_3. *Scan-based object detection* — given an image, examine a small area of the image by following a search path and determine if an object is in the area or partially included in the area. It is usually suitable for objects of a fixed size (e.g., a dot in a scatter plot, a node in a graph, or a character in a text string). It can be used to detect seeds before invoking pipeline a_2 or a starting object before invoking pipeline a_4.
  - a_4. *Tracking-based object detection* — given an image and a set of previously tracked objects, determine all pixels that belong to the next object, which may be a group of pixels along a path (e.g., a marker in a line plot), a line of pixels orthogonal to the path or an axis (e.g., a cross-section in Charles Minard's flow map [53] or a Sankey diagram).

- **Visual Object Organization**

b_1. *Relation detection and classification* — given two or more visual objects, determine whether they are related and of what type of relation (e.g., blocks forming a bar in the cholera map, two nodes of an edge in a graph, or wedges forming a pie chart).

b_2. *Object ordering* — given a set of visual objects, produce an ordered list.

b_3. *Object grouping* — given two or more visual objects and a set of relations among them (e.g., detected using b_1, construct a group object with these objects or a subset of them). In many cases, only a simple grouping procedure is needed, which requires little active learning.

b_4. *Object segmentation* — given an imagery area or a visual object, divide the area or the object into sub-objects (e.g., wedges in a pie chart following circle detection).

- **Visual Variable Measurement**

c_1. *Position measurement* — given a visual object or a group of visual objects, determine a position (x, y) (e.g., the root of a bar in the cholera map, or the center of a pie chart).

c_2. *Length measurement* — given a visual object or a group of visual objects, determine a value $length$ that represents a length measure (e.g., width, height, one of the two diagonal lengths, maximal/minimal distance between points on the contour, cross-section length, path length, or arc length).

c_3. *Area measurement* — given a visual object or a group of visual objects, determine a value $area$ that represents an area measure (e.g., bounding box area, bounding circle area, foreground area, inner circle area, or the area with a specific subset of colors).

c_4. *Angle measurement* — given a visual object or a group of visual objects, determine a value $angle$ that is measured for a wedge or curve featured in the object or group.

c_5. *Direction measurement* — given a visual object or a group of visual objects, determine a normalized or unnormalized vector (dx, dy) that is measured for the object or group.

c_6. *Scale measurement* — given a visual object or a group of visual objects, determine a mapping $scale$ for the color intensity, hue-specific color intensity, texture density, color saturation, color hue, or other options.

c_7. *Group measurement* — given a group of objects, determine a value $value$ that represents the number of visual objects in the group, or the sum, average, maximum, or minimum of the values associated with the visual objects in the group.

c_8. *Measurement transformation* — given the measured value(s) of a visual object or a group of visual objects, determine a function for transforming the visual variable(s) to data variable(s) (e.g., scaling according to an axis, applying the mapping specified by a size legend, applying logarithmic factor, applying power factor, or percentage normalization).

c_9. *Number recognition* — given a visual object or a group of visual objects, determine a value $value$ that is an integer or a real number.

c_{10}. *Text recognition* — given a visual object or a group of visual objects, determine a value $text$ that is a character or a string of characters.

The three pipelines described in this work are a_1, b_1, and c_1, and were designed, implemented, and tested in a period of two years. We can extrapolate this experience to derive an estimation that it may take in total around 20 person-years to engineer a software system with these 18 pipelines. Compared with most machine learning efforts, this pace of transforming research into a software system is considered to be relatively swift.

Tables 1 and 2 show some multi-pipeline workflows for reconstructing data from the nine historical visualizations in Fig. 12. All of these workflows can be configured with the above 18 active learning pipelines, demonstrating the potential reusability of a relatively small set of pipelines.

Table 1. Optional workflows for reconstructing data in Fig. 12(a-f), which may be configured using the 18 active learning pipelines described earlier in this section. Each workflow is denoted as $\mathbf{W}_{\text{datatype},k}$, where k indicates the k-th option. When a pipeline is given a superscript "m", "a", or "ma", it is likely most cost-effective for the pipeline to activate a purely manual, a simple algorithmic, or a parameterized algorithmic process instead of active learning. Almost all workflows will require the pipeline c_8 (measurement transformation), and thus it is not shown explicitly in the table.

Example	**Workflows**
Fig. 12(a) *Movement flow map and line plot*	$\mathbf{W}_{\text{flow},1}$: a_3^m (start cross-section) $\longrightarrow$ c_1^a (position) $\longrightarrow$ c_2 (width) $\longrightarrow$ $[\![a_1$ (stop detection (branch/merge/sharp-turn)) $\longrightarrow$ a_4 (next cross-section) $\longrightarrow$ c_1^a (position) $\longrightarrow$ c_2 (width)$]\!]$ $\longrightarrow$ b_3 (grouping all the flow segments) $\mathbf{W}_{\text{flow},2}$: a_1 (flow segment) $\longrightarrow$ b_4 (flow segment to cross-section line segments) $\longrightarrow$ c_2^a (width) $\longrightarrow$ c_1^a (position) $\longrightarrow$ b_3 (linking all the line segments) $\mathbf{W}_{\text{line},1}$: a_3^m (start point) $\longrightarrow$ c_1^a (position) $\longrightarrow$ $[\![a_4$ (next point) $\longrightarrow$ c_1^a (position)$]\!]$
Fig. 12(b) *Bar chart and line plot*	$\mathbf{W}_{\text{bar},1}$: a_3^m (start point of the top-left of the first bar) $\longrightarrow$ c_1^a (position) $\longrightarrow$ $[\![a_4$ (next point along the staircase line) $\longrightarrow$ c_1^a (position)$]\!]$ $\longrightarrow$ b_3 (points to line segments) $\longrightarrow$ c_1^a (position) $\mathbf{W}_{\text{bar},2}$: a_3^{ma} (seed per bar) $\longrightarrow$ $[\![a_4^a$ (tracking up) $\longrightarrow$ c_1^a (position)$]\!]$ $\longrightarrow$ c_8^a (uniformly redistributing points horizontally) $\mathbf{W}_{\text{line},1}$: a_3^m (start point) $\longrightarrow$ c_1^a (position) $\longrightarrow$ $[\![a_4$ (next point) $\longrightarrow$ c_1^a (position)$]\!]$ $\mathbf{W}_{\text{line},2}$: $[\![a_1$ (line segment) $\longrightarrow$ b_4^a (segment to points) $\longrightarrow$ c_1^a (position)$]\!]$ $\longrightarrow$ b_2^a (point ordering) $\mathbf{W}_{\text{line},3}$: a_1 (line segment) $\longrightarrow$ b_1^a (segment grouping) $\longrightarrow$ b_4^a (line to points) $\longrightarrow$ c_1^a (position) $\mathbf{W}_{\text{line},4}$: a_1 (cyan region) $\longrightarrow$ b_4^a (region to vertical cross-sections) $\longrightarrow$ c_2^a (cross-section length) $\mathbf{W}_{\text{line},5}$: a_3^m (seed) $\longrightarrow$ a_2 (cyan region) $\longrightarrow$ b_4^a (region to vertical cross-sections) $\longrightarrow$ c_2^a (cross-section length)
Fig. 12(c) *Thematic map with bar-chart glyphs*	$\mathbf{W}_{\text{bar},1}$: a_1 (bar) $\longrightarrow$ c_2^a (bar height) $\longrightarrow$ b_1 (bar relation) $\longrightarrow$ b_3^a (bar grouping) $\longrightarrow$ b_2^a (bar ordering) $\longrightarrow$ c_1 (glyph position) $\mathbf{W}_{\text{bar},2}$: a_1 (glyph) $\longrightarrow$ c_1 (glyph position) $\longrightarrow$ b_4 (glyph to bars) $\longrightarrow$ c_2^a (bar height) $\longrightarrow$ b_2^a (bar ordering)
Fig. 12(d) *Bubble map*	$\mathbf{W}_{\text{bubble},1}$: a_1 (bubble) $\longrightarrow$ c_3^a (bubble area) $\longrightarrow$ c_1^a (bubble position) $\mathbf{W}_{\text{bubble},2}$: a_1 (patch with number) $\longrightarrow$ c_9 (number) $\longrightarrow$ c_1 (number position)
Fig. 12(e) *Thematic map with pie-chart glyphs*	$\mathbf{W}_{\text{glyph},1}$: a_1 (wedge) $\longrightarrow$ c_3^a (wedge area) $\longrightarrow$ b_1 (wedge relation) $\longrightarrow$ b_3^a (wedge grouping) $\longrightarrow$ c_1^a (circle position) $\longrightarrow$ c_7^a (circle area) $\longrightarrow$ c_8^a (percentage normalization) $\mathbf{W}_{\text{glyph},2}$: a_1 (circle) $\longrightarrow$ c_1^a (circle position) $\longrightarrow$ c_3^a (circle area) $\longrightarrow$ b_4 (wedge segmentation) $\longrightarrow$ c_3^a (wedge area) $\longrightarrow$ c_8^a (percentage normalization) $\mathbf{W}_{\text{glyph},3}$: a_3 (seed in circle) $\longrightarrow$ a_2 (circle) $\longrightarrow$ c_1^a (circle position) $\longrightarrow$ c_3^a (circle area) $\longrightarrow$ b_4 (wedge segmentation) $\longrightarrow$ c_3^a (wedge area) $\longrightarrow$ c_8^a (percentage normalization)
Fig. 12(f) *Node-link diagram*	$\mathbf{W}_{\text{graph,bottom}}$: a_3 (node) $\longrightarrow$ a_3 (label) $\longrightarrow$ c_{10} (recognize label) $\longrightarrow$ b_1 (node with node label) $\longrightarrow$ a_4 (edge) $\longrightarrow$ b_1 (source node per edge) $\longrightarrow$ b_1 (destination node per edge) $\mathbf{W}_{\text{graph,top}}$: same as $\mathbf{W}_{\text{graph,bottom}}$ except without c_{10} (recognize label)

Table 2. Optional workflows for reconstructing data in Fig. 12(g-i), which may be configured using the 18 active learning pipelines described earlier in this section. See Table 1 for notational information.

Example	**Workflows**
Fig. 12(g) *Line plot*	$\mathbf{W}_{\text{line},1}$: a_3^m (red line start point) $\rightarrow$ c_1^a (position) $\rightarrow$ $[\![a_4$ (next point) $\rightarrow$ c_1^a (position)$]\!]$ $\rightarrow$ a_3^m (yellow line start point) $\rightarrow$ c_1^a (position) $\rightarrow$ $[\![a_4$ (next point) $\rightarrow$ c_1^a (position)$]\!]$
Fig. 12(h) *Scatter plot and line plot*	$\mathbf{W}_{\text{point},1}$: a_1 (dot) $\rightarrow$ c_1^a (dot position) $\mathbf{W}_{\text{line},1}$: a_3^m (start point) $\rightarrow$ c_1^a (position) $\rightarrow$ $[\![a_4$ (next point) $\rightarrow$ c_1^a (position)$]\!]$
Fig. 12(i) *Choropleth map*	$\mathbf{W}_{\text{region},1}$: a_3 (seed) $\rightarrow$ a_2 (region) $\rightarrow$ c_6 (texture density) $\rightarrow$ c_9 (number) $\rightarrow$ b_4^a (region to boundary points) $\rightarrow$ c_1^a (point position) $\mathbf{W}_{\text{region},2}$: a_1 (border curve segments) $\rightarrow$ b_4 (curve segment to points) $\rightarrow$ c_1^a (point position) $\rightarrow$ b_1 (curve relation) $\rightarrow$ b_3 (curves to region) $\rightarrow$ c_6 (texture density) $\rightarrow$ c_9 (number)

We can anticipate two kinds of situations, where it may be more cost-effective for a user not to use active learning for a task. In the first kind of situation, although a pipeline may offer a number of algorithmic solutions or various optional parameter settings of an algorithm, the user knows exactly which algorithmic solution or which parameter setting to use. It is more cost-effective for a user to choose the algorithm or parameter setting interactively, e.g., through a pop-up window. For example, the pipeline c_1 position measurement has the flexibility to measure the position (x, y) of a visual object by learning to recognize the signature of the position, such as top-middle, bottom-right, center of a bounding circle, or a red dot on the object. Active learning is necessary for position measurement in John Snow's cholera map. The notion of "bottom block" is not hard-coded as a predefined signature. However, for the bubbles in Fig. 12(d), the circles in (e), and the dots in (h), the user can simply select the predefined signature "center" in a pop-up window for c_1 when configuring the corresponding workflow. For such situations, we add a superscript "a" (for algorithmic) to the label of the pipeline concerned.

In the second kind of situation, there may not be enough visual objects for a pipeline to learn a model that can offer any assistance. For example, there is only one starting cross-section for the big flow segment in peach color in Fig. 12(a), and only one starting point for the line plot in Fig. 12(a), (b), and (h). It will be much quicker for a user to specify such a starting object manually. The option of manual specifications can be made available through a pop-up window for configuring the corresponding pipeline a_3 (i.e., scan-based object detection). For such situations, we add a superscript "m" (for manual) to the label of the pipeline concerned.

In some cases, a user may identify a sequence of built-in algorithms and configure an automatic or semi-automatic workflow without the need for active learning, such as $\mathbf{W}_{\text{bar},2}$ for Fig. 12(b). In other cases, although one or a few active learning pipelines are used in conjunction with built-in algorithms, the active learning becomes more or less a manual labeling process because there are not enough visual objects. $\mathbf{W}_{\text{flow},2}$ for Fig. 12(a) is such an example.

In many ways, these 18 pipelines are building blocks of most workflows for data reconstruction from historical visualizations. In a practical system, there may be a few more or fewer pipelines. Nevertheless, it is unlikely that it is necessary to have many more pipelines. It is unlikely, in the near future, an AI system can scan an arbitrary historical visualization and work out what are the visual objects that depict the required data and what may be the appropriate pipelines for the data reconstruction task. By allowing users to configure a workflow for a data reconstruction task, we enable users to use their knowledge about the semantics of a visualization and their intelligence to select a pipeline for the task. Abdul-Rahman et al. presented a visual analytics system for text similarity detection [1], which enabled users to configure a detection workflow by selecting various algorithmic modules and setting appropriate parameters for the modules. Their system provides 41 algorithmic modules and allows three parallel workflows. They reported that humanities scholars were able to learn how to configure such a workflow after two-hours of training. Based on their experience, we infer that it is feasible for users to configure a workflow with 18 optional pipelines.

8 CONCLUSION

We propose MI3, a generic pipeline for interactive classification tasks that reduces user's interaction effort. MI3 is especially useful for interactive classification applications with insufficient data to learn a precise classification model. We introduce two components of MI3 for interaction saving: algorithmic sampling and algorithmic default labeling. We demonstrate how to instantiate the generic MI3 pipeline in practice. Specifically, we demonstrate that the scenario of data reconstruction from historical spatial visualizations, which seemingly is not a classification problem, can be decomposed and transformed into three classification subtasks. The MI3 approach is shown to be suitable for all the three subtasks, and the three pipelines implemented for performing these subtasks are instantiations of the same generic approach.

Based on the decomposition, we develop a prototype software system for data reconstruction from spatial data visualizations. Its interface can be regarded as an instantiation of MI3's machine-initiated intelligent interaction for supporting active learning as well as performing classification tasks. To evaluate the usefulness of our instantiation of MI3 in the data reconstruction application, we use the simulation-based approach and compare the number of interactions needed for different solutions to accomplish the interactive classification tasks in data reconstruction. The simulation results show that MI3's algorithmic sampling and algorithmic default labeling components manage to reduce the required number of interactions. For specific subtasks of data reconstruction, an instantiation of the MI3 pipeline can save up to half of the interactions compared with a conventional ML pipeline.

There are a few limitations and related directions of this work that we hope to address in the future. From the discussion in Section 7, we can conclude that this work is a significant step towards a potential software system that can help reconstruct data from hundreds of thousands of visualization images. There is a need for using the MI3 approach to develop other pipelines listed in Section 7. Once there is a collection of ten or more pipelines, the activities for engineering a general-purpose data reconstruction software system can start.

In this work, we have only inspected one specific application scenario, namely data reconstruction, and we would like to explore other application scenarios where the MI3 approach can be effective. For various algorithmic components of MI3, including ML techniques, there is an ample scope to develop more and better solutions. For the interactive components of MI3, we have developed only one user interface as an instantiation, and we would like to explore more design options. In the evaluation, we have used a relatively simple assumption that the interaction cost is proportional to the number of interactions. The validity of this assumption will need to be investigated in future empirical studies. We have also assumed that with the termination criterion, the active learning sessions can terminate at a point that minimizes the overall interaction cost. In practice, a naive termination criterion, such as setting a fixed sample size or failure rate or giving the user full control without adequate guidance, may not achieve this goal.

REFERENCES

[1] Alfie Abdul-Rahman, Gleen Roe, Mark Olsen, Clovis Gladstone, Richard Whaling, Nicholas Cronk, Robert Morrissey, and Min Chen. 2017. Constructive Visual Analytics for Text Similarity Detection. *Computer Graphics Forum* 36, 1 (2017), 237–248.

[2] Rabah A. Al-Zaidy and Clyde Lee Giles. 2017. A Machine Learning Approach for Semantic Structuring of Scientific Charts in Scholarly Documents. In *Proceedings of the AAAI Conference on Innovative Applications (IAAI '17)*. AAAI Press, 4644–4649.

[3] Mykhaylo Andriluka, Jasper R. R. Uijlings, and Vittorio Ferrari. 2018. Fluid Annotation: A Human-Machine Collaboration Interface for Full Image Annotation. In *Proceedings of the ACM International Conference on Multimedia (MM '18)*. ACM, 1957–1966.

[4] Jürgen Bernard, Christian Ritter, David Sessler, Matthias Zeppelzauer, Jörn Kohlhammer, and Dieter Fellner. 2017. Visual-Interactive Similarity Search for Complex Objects by Example of Soccer Player Analysis. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. INSTICC, SciTePress, 75–87.

[5] Jürgen Bernard, Matthias Zeppelzauer, Michael Sedlmair, and Wolfgang Aigner. 2018. VIAL: A Unified Process for Visual Interactive Labeling. *The Visual Computer* 34, 9 (2018), 1189–1207.

[6] Jacques Bertillon. 1886. Paris 1886 Population par Groupes d'Âges et par Quartier.

[7] Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the Annual Conference on Computational Learning Theory (COLT '98)*. ACM, 92–100.

[8] Simon Bovet and Jean Bovet. 2006. GraphClick.

[9] Klaus Brinker. 2003. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of the International Conference on Machine Learning (ICML '03)*. AAAI Press, 59–66.

[10] Nicholas J. Bryan, Gautham J. Mysore, and Ge Wang. 2014. ISSE: An Interactive Source Separation Editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, 257–266.

[11] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman (Eds.). 1999. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc.

[12] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1980. The Keystroke-Level Model for User Performance Time with Interactive Systems. *Commun. ACM* 23, 7 (July 1980), 396–410.

[13] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1983. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates.

[14] Arthur Cayley. 1890. *The Collected Mathematical Papers of Arthur Cayley, Volume 3*. Cambridge University Press.

[15] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. 2002. Cluster Kernels for Semi-Supervised Learning. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS '02)*. MIT Press, 601–608.

[16] Émile Cheysson. 1906. Circulation sur les Routes Nationales, de 1851 à 1894.

[17] Ed Huai-Hsin Chi. 2000. A Taxonomy of Visualization Techniques using the Data State Reference Model. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS '00)*. IEEE, 69–75.

[18] Ed Huai-Hsin Chi and John Thomas Riedl. 1998. An Operator Interaction Framework for Visualization Systems. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS '98)*. IEEE, 63–70.

[19] Jingyu Cui, Fang Wen, Rong Xiao, Yuandong Tian, and Xiaoou Tang. 2007. EasyAlbum: An Interactive Photo Annotation System Based on Face Clustering and Re-ranking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, 367–376.

[20] Adolphe d'Angeville. 1836. Statistique de la Population Française, Population Habitans par myriametres carries en 1831.

[21] Jerry Alan Fails and Dan R. Olsen. 2003. Interactive Machine Learning. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI '03)*. ACM, 39–45.

[22] Usama Fayyad, Gregory Piatetsky-shapiro, and Padhraic Smyth. 1996. From Data Mining to Knowledge Discovery in Databases. *AI Magazine* 17, 3 (1996), 37–54.

[23] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, 29–38.

[24] Michael Friendly and Daniel J. Denis. 2001. Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualization. http://www.datavis.ca/milestones

[25] Jinglun Gao, Yin Zhou, and Kenneth E. Barner. 2012. View: Visual Information Extraction Widget for Improving Chart Images Accessibility. In *Proceedings of the IEEE International Conference on Image Processing (ICIP '12)*. IEEE, 2865–2868.

[26] Melinda T. Gervasio, Michael D. Moffitt, Martha E. Pollack, Joseph M. Taylor, and Tomas E. Uribe. 2005. Active Preference Learning for Personalized Calendar Scheduling Assistance. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI '05)*. ACM, 90–97.

[27] Arnd Gross, Sibylle Schirm, and Markus Scholz. 2014. Ycasd - A Tool for Capturing and Scaling Data from Graphical Representations. *BMC Bioinformatics* 15, 1 (2014), 219.

[28] Yuhong Guo and Dale Schuurmans. 2007. Discriminative Batch Mode Active Learning. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS '07)*. Curran Associates Inc., 593–600.

[29] Zicheng Guo and Richard W. Hall. 1989. Parallel Thinning with Two-Subiteration Algorithms. *Commun. ACM* 32, 3 (1989), 359–373.

[30] Florian Heimerl, Steffen Koch, Harald Bosch, and Thomas Ertl. 2012. Visual Classifier Training for Text Document Retrieval. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (Dec. 2012), 2839–2848.

[31] Benjamin Höferlin, Rudolf Netzel, Markus Höferlin, Daniel Weiskopf, and Gunther Heidemann. 2012. Inter-Active Learning of Ad-Hoc Classifiers for Video Visual Analytics. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST '12)*. IEEE, 23–32.

[32] Thomas S. Huang, Charlie K. Dagli, Shyamsundar Rajaram, Edward Y. Chang, Michael I. Mandel, Graham E. Poliner, and Daniel P. W. Ellis. 2008. Active Learning for Interactive Multimedia Retrieval. *Proc. IEEE* 96, 4 (April 2008), 648–667.

[33] Weihua Huang and Chew Lim Tan. 2007. A System for Understanding Imaged Infographics and Its Applications. In *Proceedings of the ACM Symposium on Document Engineering (DocEng '07)*. ACM, 9–18.

[34] Thorsten Joachims. 1999. Transductive Inference for Text Classification Using Support Vector Machines. In *Proceedings of the International Conference on Machine Learning (ICML '99)*. Morgan Kaufmann Publishers Inc., 200–209.

[35] Bonnie Elizabeth John. 1988. *Contributions to Engineering Models of Human-computer Interaction.* Ph.D. Dissertation. Carnegie Mellon University.

[36] Ajay J. Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. 2009. Multi-Class Active Learning for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*. IEEE, 2372–2379.

[37] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. 2017. ChartSense: Interactive Data Extraction from Chart Images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, 6706–6717.

[38] Saurabh Kataria, William Browuer, Prasenjit Mitra, and Clyde Lee Giles. 2008. Automatic Extraction of Data Points and Text Blocks from 2-dimensional Plots in Digital Documents. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI '08)*. AAAI Press, 1169–1174.

[39] David E. Kieras. 1988. Towards a Practical GOMS Model Methodology for User Interface Design. In *Handbook of Human-Computer Interaction*, Martin G. Helander (Ed.). Elsevier, 135–157.

[40] Klokan Technologies GmbH. Accessed: 2020-05-31. Old Maps Online. https://www.oldmapsonline.org/

[41] Kostiantyn Kucher, Carita Paradis, Magnus Sahlgren, and Andreas Kerren. 2017. Active Learning and Visual Analytics for Stance Classification with ALVA. *ACM Transactions on Interactive Intelligent Systems* 7, 3, Article 14 (Oct. 2017), 31 pages.

[42] Branislav Kveton and Shlomo Berkovsky. 2016. Minimal Interaction Content Discovery in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems* 6, 2, Article 15 (2016), 25 pages.

[43] Christian Leistner, Amir Saffari, Jakob Santner, and Horst Bischof. 2009. Semi-Supervised Random Forests. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV '09)*. IEEE, 506–513.

[44] David D. Lewis and Jason Catlett. 1994. Heterogeneous Uncertainty Sampling for Supervised Learning. In *Proceedings of the International Conference on Machine Learning (ICML '94)*. Morgan Kaufmann Publishers Inc., 148–156.

[45] Hongsen Liao, Li Chen, Yibo Song, and Hao Ming. 2016. Visualization-Based Active Learning for Video Annotation. *IEEE Transactions on Multimedia* 18, 11 (Nov. 2016), 2196–2205.

[46] Library of Congress. Accessed: 2020-05-31. Library of Congress Digital Collections. https://www.loc.gov/collections/

[47] Dong Liu, Meng Wang, Xian-Sheng Hua, and Hong-Jiang Zhang. 2009. Smart Batch Tagging of Photo Albums. In *Proceedings of the ACM International Conference on Multimedia (MM '09)*. ACM, 809–812.

[48] Ruizhe Liu, Weihua Huang, and Chew Lim Tan. 2007. Extraction of Vectorized Graphical Information from Scientific Chart Images. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR '07, Vol. 1)*. IEEE, 521–525.

[49] Yiwen Luo, Wei Liu, Jianzhuang Liu, and Xiaoou Tang. 2008. MQSearch: Image Search by Multi-Class Query. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, 49–52.

[50] Gonzalo Gabriel Méndez, Miguel A. Nacenta, and Sebastien Vandenheste. 2016. iVoLVER: Interactive Visual Language for Visualization Extraction and Reconstruction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, 4073–4085.

[51] Charles Joseph Minard. 1845. Carte de la Circulation des Voyageurs par Voitures Publiques sur les routes de la contrée où sera placé le Chemin de Fer de Dijon à Mulhouse.

[52] Charles Joseph Minard. 1859. Carte Figurative et approximative des tonnages des Grand Ports et des principales Rivières d'Europe.

[53] Charles Joseph Minard. 1869. Carte figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813.

[54] Hieu T. Nguyen and Arnold Smeulders. 2004. Active Learning Using Pre-clustering. In *Proceedings of the International Conference on Machine Learning (ICML '04)*. ACM, 79.

[55] Florence Nightingale. 1858. *Mortality of the British Army: At Home and Abroad, and During the Russian War, as Compared with the Mortality of the Civil Population in England.* London: Harrison and Sons.

[56] Florence Nightingale. 1858. *Notes on Matters Affecting the Health, Efficiency, and Hospital Administration of the British Army.* London: Harrison and Sons.

[57] Alban William Housego Phillips. 1958. The Relation between Unemployment and the Rate of Change of Money Wage Rates in the United Kingdom, 1861-1957. *Economica* 25, 100 (1958), 283–299.

[58] William Playfair. 1786. *The Commercial and Political Atlas: Representing, by Means of Stained Copper-Plate Charts, the Progress of the Commerce, Revenues, Expenditure, and Debts of England, during the Whole of the Eighteenth Century.* London: Debrett, Robinson, and Sewell.

[59] William Playfair. 1786. Exports & Imports to and from all North-America.

[60] William Playfair. 1801. *The Statistical Breviary: Shewing, on a Principle Entirely New, the Resources of Every State and Kingdom in Europe.* Wallis, London, UK.

[61] William Playfair. 1822. Chart Shewing at One View the Price of the Quarter of Wheat & Wages of Labour by the Week from the Year 1565 to 1821.

[62] Jorge Poco and Jeffrey Heer. 2017. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Computer Graphics Forum* 36, 3 (June 2017), 353–363.

[63] Jorge Poco, Angela Mayhua, and Jeffrey Heer. 2018. Extracting and Retargeting Color Mappings from Bitmap Images of Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 637–646.

[64] Dimitrios Rafailidis, Apostolos Axenopoulos, Jonas Etzold, Stavroula Manolopoulou, and Petros Daras. 2014. Content-Based Tag Propagation and Tensor Factorization for Personalized Item Recommendation Based on Social Tagging. *ACM Transactions on Interactive Intelligent Systems* 3, 4, Article 26 (Jan. 2014), 27 pages.

[65] Ankit Rohatgi. 2011. WebPlotDigitizer. https://automeris.io/WebPlotDigitizer

[66] David Rumsey. Accessed: 2020-05-31. David Rumsey Map Collection. https://www.davidrumsey.com/

[67] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. 2008. LabelMe: A Database and Web-Based Tool for Image Annotation. *International Journal of Computer Vision* 77, 1-3 (2008), 157–173.

[68] Dominik Sacha, Matthias Kraus, Daniel A. Keim, and Min Chen. 2019. VIS4ML: An Ontology for Visual Analytics Assisted Machine Learning. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 385–395.

[69] Dominik Sacha, Michael Sedlmair, Leishi Zhang, John A. Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C. North, and Daniel A. Keim. 2017. What You See Is What You Can Change: Human-Centered Machine Learning by Interactive Visualization. *Neurocomputing* 268 (Dec. 2017), 164–175.

[70] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. 2011. ReVision: Automated Classification, Analysis and Redesign of Chart Images. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, 393–402.

[71] Christin Seifert and Michael Granitzer. 2010. User-Based Active Learning. In *IEEE International Conference on Data Mining Workshops*. IEEE, 418–425.

[72] Burr Settles. 2009. *Active Learning Literature Survey.* Technical Report. University of Wisconsin–Madison.

[73] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. 2016. FigureSeer: Parsing Result-Figures in Research Papers. In *Proceedings of the European Conference on Computer Vision (ECCV '16)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, 664–680.

[74] John Snow. 1854. Cholera Map.

[75] John Snow. 1855. *On the Mode of Communication of Cholera.* London: John Churchill.

[76] Qixiang Tan (Ed.). 1982. *The Historical Atlas of China.* China Cartographic Publishing House.

[77] Jinhui Tang, Qiang Chen, Meng Wang, Shuicheng Yan, Tat-Seng Chua, and Ramesh Jain. 2013. Towards Optimizing Human Labeling for Interactive Image Tagging. *ACM Transactions on Multimedia Computing, Communications, and Applications* 9, 4, Article 29 (Aug. 2013), 18 pages.

[78] Yuandong Tian, Wei Liu, Rong Xiao, Fang Wen, and Xiaoou Tang. 2007. A Face Annotation Framework with Partial Clustering and Interactive Labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*. IEEE, 1–8.

[79] Wei Tong and Rong Jin. 2007. Semi-Supervised Learning by Mixed Label Propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI '07)*. AAAI Press, 651–656.

[80] Bas Tummers. 2006. DataThief III. http://datathief.org/

[81] United States Census Bureau. Accessed: 2020-05-31. United States Census Bureau Publications. https://www.census.gov/library/publications.html

[82] Jarke J. van Wijk. 2005. The Value of Visualization. In *Proceedings of the IEEE Conference on Visualization (VIS '05)*. IEEE, 79–86.

[83] Julie S. Weber and Martha E. Pollack. 2007. Entropy-Driven Online Active Learning for Interactive Calendar Management. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI '07)*. ACM, 141–150.

[84] Zuobing Xu, Ram Akella, and Yi Zhang. 2007. Incorporating Diversity and Density in Active Learning for Relevance Feedback. In *Proceedings of the European Conference on IR Research (ECIR '07)*. Springer-Verlag, 246–257.

[85] David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL '95)*. Association for Computational Linguistics, 189–196.

[86] Yu Zhang, Bob Coecke, and Min Chen. 2019. On the Cost of Interactions in Interactive Visual Machine Learning. In *Proceedings of the IEEE VIS Workshop on Evaluation of Interactive Visual Machine Learning Systems*. IEEE, 5.

[87] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with Local and Global Consistency. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS '03)*. MIT Press, 321–328.

[88] Yan Ping Zhou and Chew Lim Tan. 2000. Hough Technique for Bar Charts Detection and Recognition in Document Images. In *Proceedings of the International Conference on Image Processing (ICIP '00, Vol. 2)*. IEEE, 605–608.

[89] Xiaojin Zhu and Zoubin Ghahramani. 2002. *Learning from Labeled and Unlabeled Data with Label Propagation*. Technical Report. Carnegie Mellon University.

A BLOCK NEIGHBORHOOD FEATURES

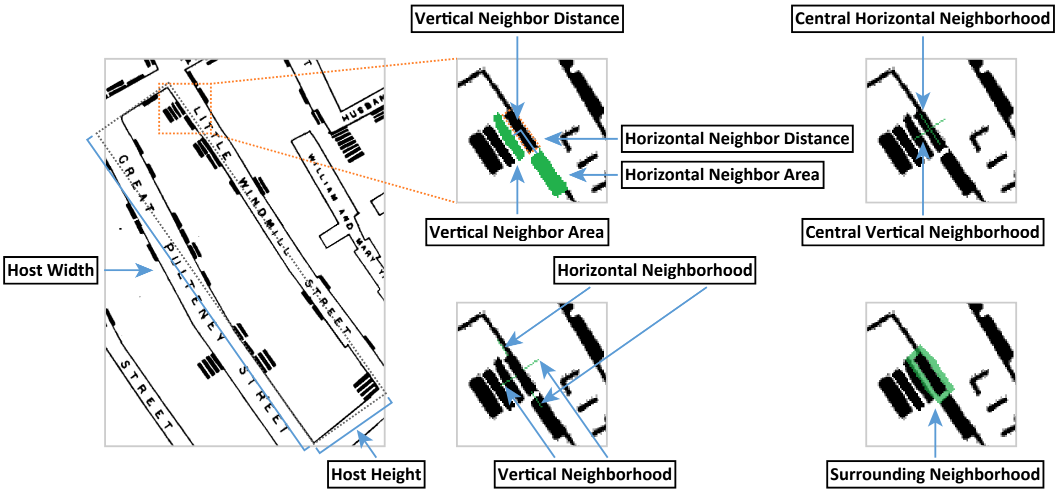


Fig. 13. Neighborhood features and measurements. Horizontal/Vertical neighborhood refers to the union of 10 pixels horizontally/vertically extending outward the candidate block along its horizontal/vertical axis. The measurements of central horizontal and vertical neighborhoods are similar. Surrounding neighborhood refers to the union of pixels within 5 pixels' distance from the border of the candidate block.

In some cases, true detections and false detections of blocks are not differentiable by appearance. For example, dashes in dashed lines and blocks look identical in the cholera map. To handle such cases, we compute neighborhood features to capture the image features in the candidate blocks' neighborhood. Fig. 13 illustrates some of the neighborhood features and neighborhood measurements used for feature computation. For conciseness, we refer to the two symmetry axes of a candidate block's bounding box as the candidate block's horizontal and vertical axis in the following. The horizontal/vertical axis is the one extending along the long/short edge of the bounding box.

Firstly, we compute neighbor distribution features to depict the distribution of other connected components around the candidate block. The *horizontal/vertical neighbor distance* captures the distance to the nearest component (with the background excluded) in the horizontal/vertical direction. The *surrounding neighbor number* feature denotes the number of adjacent components to the candidate block.

Secondly, We compute neighbor features to capture the properties of neighboring connected components of the candidate block. *Horizontal/Vertical neighbor area* measures the area of the closest neighboring connected component (with the background excluded) in the horizontal/vertical direction. *Host width*, *host height*, *host area*, and *host solidity* measure the width, height, area, and solidity of the connected components containing the candidate block.

Thirdly, we compute neighborhood foreground rate features to capture the distribution of foreground pixels (i.e., white pixels in the binarized image) in the candidate block's neighborhood. *Surrounding foreground rate* is computed as the rate of foreground pixels within 5 pixels' distance from the candidate block's border. *Horizontal/Vertical foreground rate* is computed as the rate of foreground pixels within 10 pixels' distance from the candidate block's border along the horizontal/vertical axis. Similarly, *central horizontal/vertical foreground rate* measures the rate of foreground pixels along the two axes, but concern the pixels within 10 pixels' distance from the candidate block's center instead of the border.

B ROOT NEIGHBORHOOD FEATURES

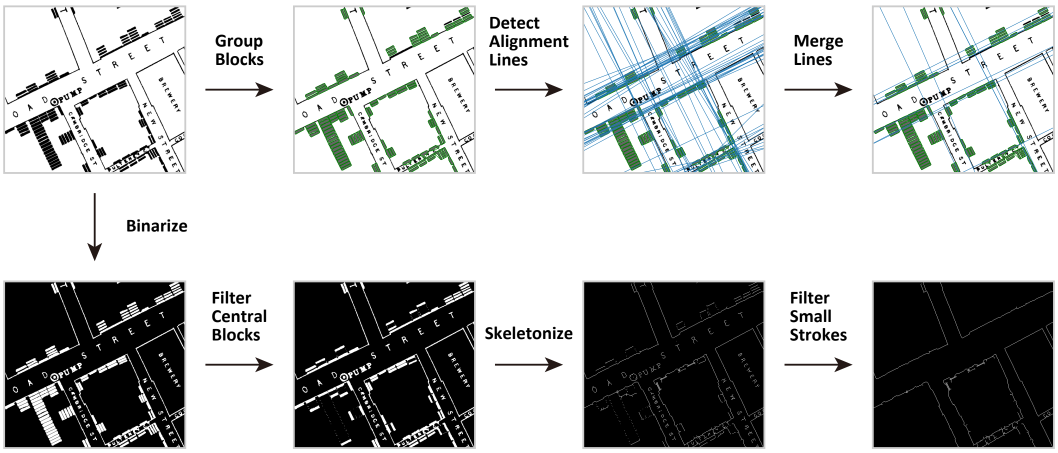


Fig. 14. The first row shows the process of detecting alignment lines, which is used for computing the distances from candidate roots to alignment lines. The second row shows the process of detecting the skeletons, which is used for computing the distances from candidate roots to skeletons.

Distance to alignment line captures how well the candidate root aligns with other candidate roots. For each candidate root, we denote the horizontal axis of the block containing the root as the *alignment line* that it generates. For all the alignment lines generated by all the candidate roots, we merge the alignment lines that are close to each other. Quantitatively, we denote the distance between two lines to be the maximal distance from a point on a line within the image range to the other line. Two lines are merged when their distance is smaller than the minimum width and height of blocks. For the merged lines, we refit a line with linear regression of the positions of the roots that generate the lines. Then, we filter the lines that cannot be merged with any other lines. This process of computing the alignment lines is shown in the first row of Fig. 14. With the merged lines (Fig. 14 top right), we can measure the *distance to alignment line* of each candidate root.

Distance to skeleton captures whether the candidate root is close to a boundary line in the image. As shown in the second row of Fig. 14, to compute this feature, we detect the skeleton of the binarized image using Guo and Hall's thinning algorithm [29]. To remove noise for skeleton detection, we further adopt two strategies. Firstly, we remove all the pixels occupied by blocks that do not contain candidate roots before the skeleton detection. In this way, these pixels will not be mistaken for the skeleton. Secondly, after the skeleton detection, we filter the detected skeleton strokes that are too small. We compute the connected components formed by the skeleton pixels using 8-connectivity and remove the connected components whose diagonal of the axis-aligned bounding box is shorter than twice the maximal width and height of blocks. Using the filtered skeleton, we measure the *distance to skeleton* of each candidate root.