Integrating Visualization with Original Data

Category: Research

Paper Type: please specify

Fig. 1. In the Clouds: Vancouver from Cypress Mountain. Note that the teaser may not be wider than the abstract block.

Abstract—Visualizations play an important role in the display and communication of knowledge. When sharing visualization through images, it is also desirable to provide users with the original data. In this work, we propose a method for integrating source data with target visualization images through QR Codes. Users can retrieve the original data from the visualization image and even restore the visualization and make further explorations.

Index Terms—Visualization Data

1 Introduction

Visualizations convey information by mapping data to the visual attributes of graphical marks [1]. As the most commonly available format for visualizations [11, 12], bitmap images are convenient to read, edit, and transfer among different users in real-world applications, such as digital documents, online news, social media. Many visualization design tools support to export visualizations as bitmap images, including Excel, Google Spreadsheets, Tableau/Polaris [29], and Lyra [25]. The visualization systems based on programmatic toolkits or libraries (e.g., D3 [3], Vega-Lite [26], ProtoVis [2]) also usually provide the function of saving the data exploration status as bitmap images.

As the visualization images are shared among people, the detailed information of visualizations (e.g., underlying data, visualization method, visualization parameters) are always lost. There are many reasons a viewer wants to extract these informations from visualization bitmap images. For instance, a visualization image shows the topological structure of a multivariate hierarchical dataset and a viewer might wish to further explore the relationships between topological structure and attribute values. A viewer might prefer to view an adjacent matrix in the form of force-directed layout, which is easier to learn the overview of the graph. A viewer might need to restore the exploration status of an interactive visualization system in the bitmap image for interactive analysis. However, bitmap images only show the visualization results after data transformation, visual mapping, and interactive exploration [TODO]. It is difficult for viewers to retrieve the desired information efficiently and accurately.

Many research studies have applied computer vision techniques for extracting underlying data or visualization methods from bitmap images. The methods which extract the underlying data can be classified into three categories, interactive approach [19], automatic approach [8, 14, 16, 27], and mix-initiative approach [17, 24, 30]. However, these techniques can only support the data extraction from simple chart images in specific types, for example, bar chart, line chart, pie chart. Many popular visualizations like graphs and trees cannot be supported. Secondly, the extracted data is only based on the graphical marks in the bitmap images. It means that if the images only show the visual elements after filtering, the extracted results are also the corresponding subset instead of the whole dataset. Thirdly, the data extraction accuracy depends largely on edge detection or vectorization algorithms [17]. Therefore, the noisy image processing results will degrade the accuracy significantly. Interactive manipulation could improve accuracy, but it can make the data extraction process tedious and time-consuming. In addition to extracting underlying data, Poco and Heer [21] proposed reverse-engineering visualizations, which can recover visual encodings from bitmap images using inferred text elements. However, this method is also limited to simple chart images.

We propose a method to integrate the bitmap images with visualization information (e.g., original data, visualization method, parameters) together, which can support viewers to retrieve the desired information from visualization images efficiently and accurately. Our method is inspired by the information hiding techniques, which are designed to hide secret messages inside another in a way that the hidden message is imperceptible to viewers. The imperceptibility of the information hiding technique guarantees that the images after embedding data will not influence users' perception of original visualizations.

Information hiding methods mainly include steganography and digital watermarking. Steganography methods emphasize the imperceptibility of the embedded data and have a relatively high capability. However, they are usually not robust to attack. Digital watermarking methods are more robust against various attacks, but the capacity is usually extremely limited. In addition, the embedded results often suffer from

image distortions [18]. Visualization bitmap images have very different characteristics from photographs. Firstly, the underlying data might be large, ranging from several kilobytes to several megabytes. Secondly, different from the attacks against the digital watermarking methods, such as rotation and cropping [28], the modifications on the visualization images are usually the editing of users, for example, users mark the interesting parts in the visualization images. As a result, hiding information into visualization bitmap images is a challenging task, because it requires not only imperceptibility and recoverability but also robustness and high capability

Our method embeds data by modifying the least-significant-bit (LSB) of host images to achieve high capability. To improve data extraction robustness against users' editing, we introduce QR code ("Quick Response" code) during the process of information hiding. Specifically, we convert the information into QR codes at first and then hide the QR codes into the visualization images. QR code is a 2D matrix code that is designed to store a large amount of data, which can be damage resistant and decoded at high speed. We use the integer programming method to compute the optimal parameters of QR code to encode data. The arrangements of pixels in the QR code are also designed to resist users' editing. We developed an online interactive system to hide/extract the information into/from visualization bitmap images. We also developed a python library, which can support the data hiding/extracting process locally.

We validate our data hiding/extracting method through the real-world applications of visualization bitmap images, including the digital documents and online news. The validations are from three aspects. Firstly, the result shows that the visualization images with hiding information do not influence users' perception. Secondly, the capability of our method meets the requirements of most visualizations. Thirdly, after users' editing in the visualization images, our method can still extract the hidden information.

Our contributions are, (1) integrating the information into visualization bitmap images through changing the LSBs without influencing users' perception; (c) improving the robustness to resist users' editing in images through using QR code; (3) developing an online interactive system and a local python library for information hiding and extracting; (4) validating the utility of our information hiding methods through two real-world application scenarios.

2 Related Work

This section introduces prior studies that are most relevant to our work, including data extraction from visualization images and information hiding techniques.

2.1 Data Extraction

Creating visualizations needs to map data to the visual attributes (e.g., position, area, color) of graphical marks; How to deconstruct a visualization to extract its underlying data, marks, and visual mappings has been one vital research direction in the visualization community.

The visualizations accessible to the masses are classified into two categories, bitmap-based and SVG-based. Bitmap-based visualization images are the most commonly available format in real-world applications [12], such as online news, digital documents, and social media. The visual elements in visualizations bitmap images are composed of several individual pixels. As a result, research studies always extract the high-level shapes (e.g., line segments) from the visualization images at first based on image processing and machine learning techniques, and then classify different visualizations using these shapes as features [15, 31, 32].

In addition to classification, many research studies [14–16, 32] have investigated techniques for extracting marks from charts based on image processing algorithms. However, these methods only focus on extracting the graphical marks rather than the underlying data; Their extraction accuracy relies heavily on the edge detection results, which are difficult to retrieve from real-world visualization bitmap images accurately because of the variant image qualities.

The methods to extract underlying data are classified into three categories, automatic, mix-initiative, and interactive approaches. Savva et al. proposed an *automatic* pipeline in ReVision [27], which exploits the machine learning method to classify charts, and then extracts the graphical marks and infers the underlying data. The classification accuracy of ReVision is about 80% on average, and it extracts marks successfully from 71% of bar charts and 64% of pie charts. On the basis of ReVision's twofold pipeline, ChartSense [17] utilize a *mixed-initiative* approach to improve the extraction accuracy. It applies an interactive data extraction algorithm according to the chart types and provides a set of simple interactions to fine-tune the results. The mix-initiative methods also include WebPlotDigitizer [24] and DataThief [30], which extract underlying data of chart images using both automatic algorithm and manual specifications. Ycasd [9] and iVoLVER [19] belong to the *interactive* category, which relies on manual input. For example, Ycasd [9] requires users to specify all the data points in a line chart and iVoLVER [19] requires to specify the data types (e.g., text, colors, shapes, etc.) and sampling data points in the visualization images. Interactive methods can guarantee the extraction accuracy, but the user manipulations make the data extraction process tedious and time-consuming. Note that all the data extraction methods above are only suitable for chart images (e.g., bar chart, line chart, pie chart), some visualizations (e.g., graph, tree) are also very popular but cannot be supported.

Besides the underlying data, research studies also extract visualization parameters or templates from images. Poco and Heer [21] propose reverse-engineering visualizations pipeline to extract visual encodings from a chart image. The pipeline first detects the text element within bitmap chart images, then classifies their role, and finally recovers the text content using optical character recognition. Based on the the same extraction pipeline with reverse-engineering techniques, Poco, Mayhua, and Heer [22] use the mix-initiative model to extract color mappings, which allows users to correct the interpretation errors using an annotation interface. Different with the above extraction pipeline, Chen et al. [4] propose a multi-task deep neural network to deconstruct the timeline infographics into global and local information and reconstruct extensible templates. Existing methods can only be used to extract specific parameters (e.g., color mapping, visual mapping) from restricted visualization images (e.g., chart images, timeline infographics).

Different from the bitmap-based visualizations, the SVG-based visualizations are composed of vector graphics, including circle, rectangle, line. Existing research studies focus on D3 [3] visualizations because they are ubiquitous and based on open Web standards (HTML, CSS, SVG, etc.) that allow inspection. Harper and Agrawala [11] developed a technique for deconstructing existing SVG-based D3 visualizations to extract the data, the marks, and the mappings between them and restyling. While they provide a graphical interface for interactively restyling D3 charts, their tool is primarily aimed at visualization experts. Therefore, Harper and Agrawala [12] continue to introduce an algorithm for converting a basic D3 chart into a reusable style template, which can be applied to a new data source directly to produce new visualizations directly. In addition to extracting the underlying data, Lu et al. [20] propose Interaction+ to facilitate the explorations, which extracts the visual marks from existing SVG-based visualizations and augment them by providing a suite of interactions such as filtering, comparison, and annotation.

2.2 Information Hiding

Information hiding techniques have a long history for hundreds of years [7]. With the increasing use of the computes, the information hiding techniques are applied to hide secret messages within the computer files such as texts, images, audio, and videos so that no one except the sender and the receiver will doubt the existence of hidden information in it.

Information hiding techniques are mainly divided into two categories, steganography, and digital watermarking [28]. Steganography emphasizes the imperceptibility of hidden information. The most common approach of steganography technique embeds information into host files by modifying the least significant bits (LSBs). Therefore, steganography has relatively high capacity, but are not robust to attacks. Except modifying the LSBs, Hao, Feng, and Zhou [10] propose a tech-

nique to hide binary images into the gradient domain of host color image. This method modifies the gradient vectors as a chosen hidden vector orientation in CIELAB color space. The modifications of pixel values are restricted within the just-noticeable difference (JND) to ensure the imperceptibility of hidden information. However, these above information hiding methods cannot resist users' editing on visualization images.

Digital watermarking is mainly used to protect the ownership rights (e.g., the textual information about the author and copyright) in digital form under transmission or transformation. As a result, digital watermarking focuses on the recoverability of information, but their capacity is usually extremely limited. Hota and Huang [13] proposed self-describing visualizations, which embed the meta-information into visualization images. Self-describing visualizations use the digital watermarking methods because the provenance information embedded in the visualization images is very small. However, the digital watermarking method cannot be used to embed the hundreds of kilobytes of underlying data into visualization images.

Despite the substantial previous works on data extraction and information hiding, to the best of our knowledge, previous methods are not able to allow users to extract/hide the underlying data with the visualization images.

3 Method

To human, a visualization image shows a collection of visual elements, the visual attributes (color, positions) of which encodes the data values. However, to a computer, an image is an array of numbers that represent color at various points (pixels) as shown in Figure 2. For example, a common image size with 640×480 pixels and 256 colors (8 bits per pixel) contain about 250 kilobits of data.

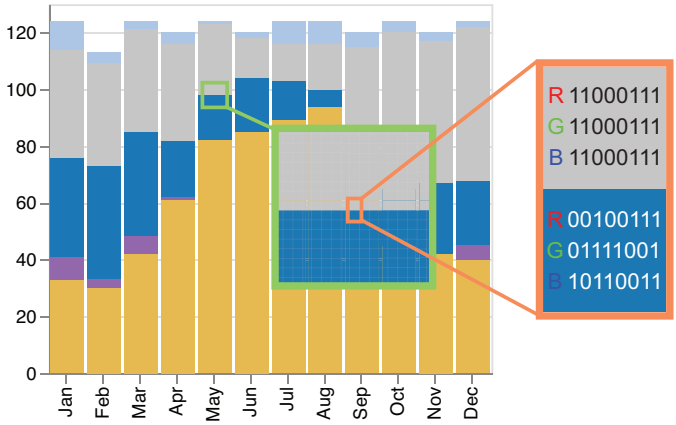

Fig. 2. The left part of the figure above shows that the visualization images are constituted by various pixels with different color. The right part shows that the color value of each pixel are represented by 24 bits (3 bytes). The visualization above is available at `https://vega.github.io/vega-lite/examples/stacked_bar_weather.html`

Existing methods [] extract the underlying data from human's perspective based on the image processing techniques. On the contrary, our method is from computers' perspective, which regards the visualization image as the collection of numbers and hides/extracts information from these numbers.

Figure 10 shows the overview pipeline of the information hiding and extracting methods based on QR-code. The technical details are introduced below.

3.1 Steganography

Steganography is the art of hiding information in the cover object, which can be a digital medium such as image, audio or video file, to obtain a stego file that has secret information hidden in it [6]. Specifically, the cover object of our method is the visualization images. In image steganography, the information is hidden exclusively in images. The image steganography can be implemented through different methods such as methods of substitution of LSB, transform space statistical and distortion techniques, etc. Here our method is based on the least-significant-bit technique for image steganography.

The least-significant-bit technique is a kind of substitution algorithm, which embed data by substituting carefully chosen bits from the cover image pixels with secret message bits. A message could be plain text, ciphertext, other images, or anything that can be embedded in a bit stream [6]. The technique involves the modification of the LSB planes of the image. In this technique, the message is stored in the LSB of the pixels which could be considered as random noise. Therefore altering them does not significant affect the quality of the cover image. The variation of the LSB algorithm include one or more LSB bits to be changed to a bit of secret message.

(00100111 11101001 11001000) (00100111 11101000 11001000)
(00100111 11001000 11101001) → (00100110 11001000 11101000)
(11001000 00100111 11101001) (11001000 00100111 11101001)

Fig. 3. The left part shows the RGB binary value of three pixels. The right part shows the results after hiding character A (10000011). The underlined bits in the right part construct the binary value of character A. The bits in red actually changed during the information hiding process.

To hide an image in the LSBs of each byte of a 24-bit image, each pixel can store 3 bits. Therefore a 640×480 image has the potential to hide a total of 921,600 bits (115,200 bytes) of information. To the human eye, the resulting stego-image will look identical to the original cover image. Figure 3 shows the example of inserting the character A in three pixels (assuming no compression) with only three bits actually changed. Existing studies [23] show that LSB technique only change half the bits in an image. In addition, existing studies also validate that users can hide data in the least and second least significant bits of each byte with guaranteeing the imperceptibility of hidden data [**?**].

However, the method of hiding the data into the image directly is vulnerable. For example, if compress the size of images or manipulate the some pixels of images even slightly, users cannot extract the hidden information.

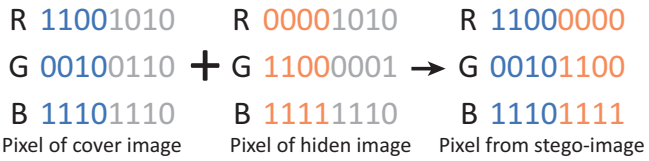

Fig. 4. Steganography technique of hiding image. The pixel from stego-image keeps the most significant bits of the pixel from hidden images and cover image.

Besides the plain text, steganography technique can also hide image inside another. The difference between hiding the plain text is that it requires to manipulate more bits of the pixel values to store the data of hidden image. For each RGB binary value (8-bit), the rightmost bit is the less significant bit. On the contrary, the leftmost bit is the most significant bit. To guarantee the less impact on the pixel values of cover images and hidden images and, this technique changes the less significant bits from cover images and include the most significant bits of hidden images as shown in Figure 4.

3.2 QR Code

QR code [5] is invented in 1994 by Denso Wave Incorporated, Japan. QR code encode the information in two directions: horizontally and vertically. QR code can be read easily for machine. In addition, the most important characteristic of QR code is that it is capable of holding a great deal of information. Specifically, QR code can manage all types of data, including letters, numbers, images. Figure 6 shows that the capacity of a QR code that encodes 524 alphanumeric characters. The capabilities of QR code are determined by its size, error correction

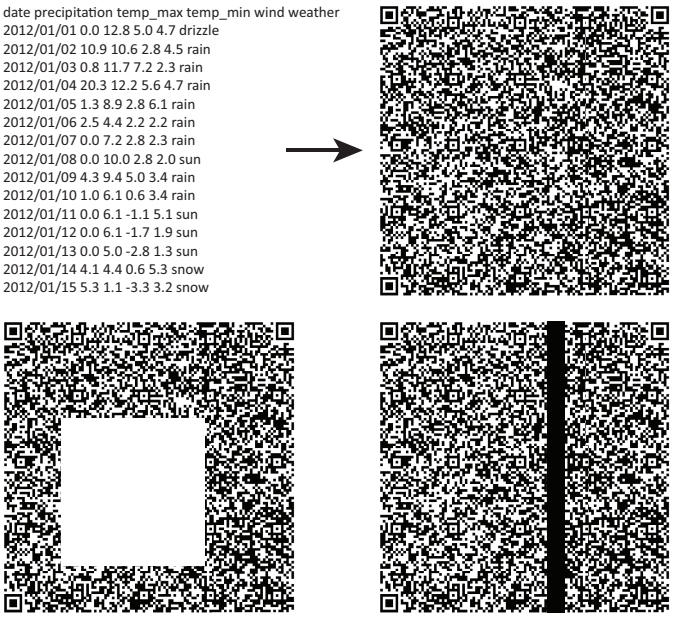

Fig. 5. The top part of the figure above shows that a QR-code can encode 524 alphanumeric characters. The bottom part are the damaged or dirty QR-code.

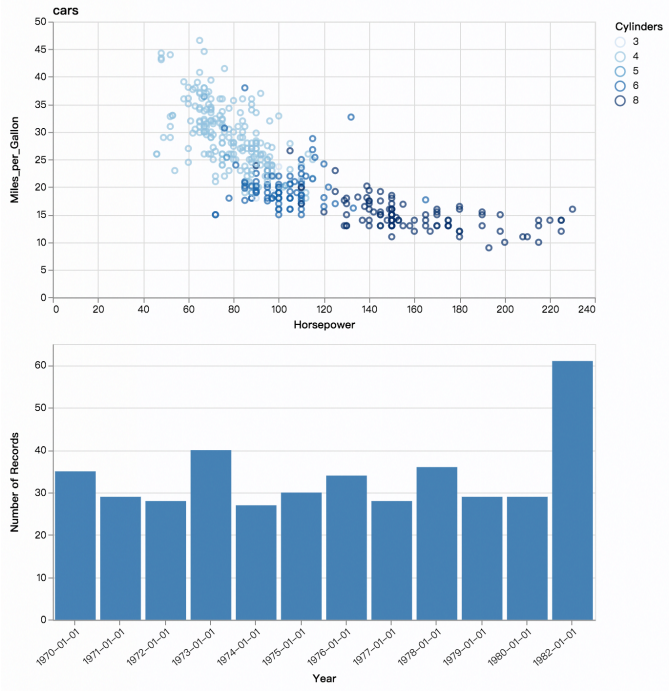

Fig. 6. The top part of the figure above shows that a QR-code can encode 524 alphanumeric characters. The bottom part are the damaged or dirty QR-code.

levels (L, M, Q, H), and data types of hidden information (Numeric, Alphanumeric, etc.). One QR code can encode up to 4,296 alphanumeric characters with the maximum size and the lowest error correction level. The size of QR-code is incremented by four cells in both vertical and horizontal direction - 21 $\times$ 21 cells, 25 $\times$ 25 cells, 29 $\times$ 29 cells, and there are 40 size types with the maximum size set to 177 $\times$ 177 cells. The QR code has the ability to perform error correction, which means that data in QR codes can be recovered even if parts of the symbol have been destroyed or damaged. Figure 6 also shows that the data encoded in the partially damaged or dirty QR-code can still be restored.

3.3 Hiding Method

To improve the robustness of data hiding/extracting, our method does not hide the text-format original data into the visualization images directly, but transform the original data to QR-code at first and then hide the QR-code instead.

Different from the ordinary images, the QR-code image is a binary image. Storing each pixel of the QR-code image only need one bit, specifically, one for the pixel in white and zero for the pixel in black. Therefore, hiding each pixel in one QR-code image only need to modify one bit of the pixel values in the visualization image. Because the least and second least significant bits of each byte can be modified to hide data [**?**], each pixel in the visualization image can hide up to six QR-code images. According to the information capabilities of QR-code [5], multiple QR-codes have much less storage than a single QR-code with occupying the same space. A 640$\times$480 visualization image can contain 36 version-40 QR-codes (177$\times$177 pixels), 30 version-27 QR-codes (125$\times$125 pixels), 18 version-23 QR-codes (109$\times$109). These QR-codes can encode up to 247,200 alphanumeric characters.

To further improve the stability, each cell in the QR-code image could occupy more than one pixel. As shown in Figure **??**, one cell in the QR-code image match with four pixels in the visualization images. Although this method reduces the storage of hiding data, it prevents the impacts of small manipulations for visualization images (e.g., adding some random noise to the resulting images).

The data hidden in the visualization images are divided into three categories, underlying data, visualization specifications, and exploration records. The underlying data allows users to restyle or change the original visualizations. The visualization specifications determine the visual mappings from the original data to visual elements and the interaction designs. The exploration records help users' further exploration based on previous findings.

- **Underlying data**. Some informations from the original data are lost after mapping into the visual elements. The original dataset is necessary for providing users with the comprehensive understanding. QR-code could store the underlying data in multiple data formats (e.g., json, csv, etc.).

- **Visualization method**. Our method supports to two ways to programmatically define a visualization. The first is to use imperative languages and libraries, which support the construction of visualizations from the beginning, including Prefuse, D3, Processing and Protovis. The second way is to use a declarative language, which is also widely used by the visualization community. By decoupling the specification from the execution, declarative visualization grammars allow users to specify what is shown in visualizations directly. Firstly, providing the visualization method for users allows to restore the visualizations automatically and explore the data interactively. Secondly, the For example, changing the visualization categories may improve the visualization and modifying the data encodings can restyle the visualizations.

- **Exploration records**: Both of these two kinds of data mentioned above focus on the visual elements in the visualizations, the one is the underlying data of the visual elements and the other is visual mapping. Besides these data, exploration processes could also be integrated into visualization images. Retrieving these data supports users to make further explorations based on the findings of previous users.

3.3.1 Data Size

Steganography can hide the string, image and etc into the host image.
compute the capability of the pure data
compute the capability of the data through QR code
and compare their capabilities
and further improve its capabilities

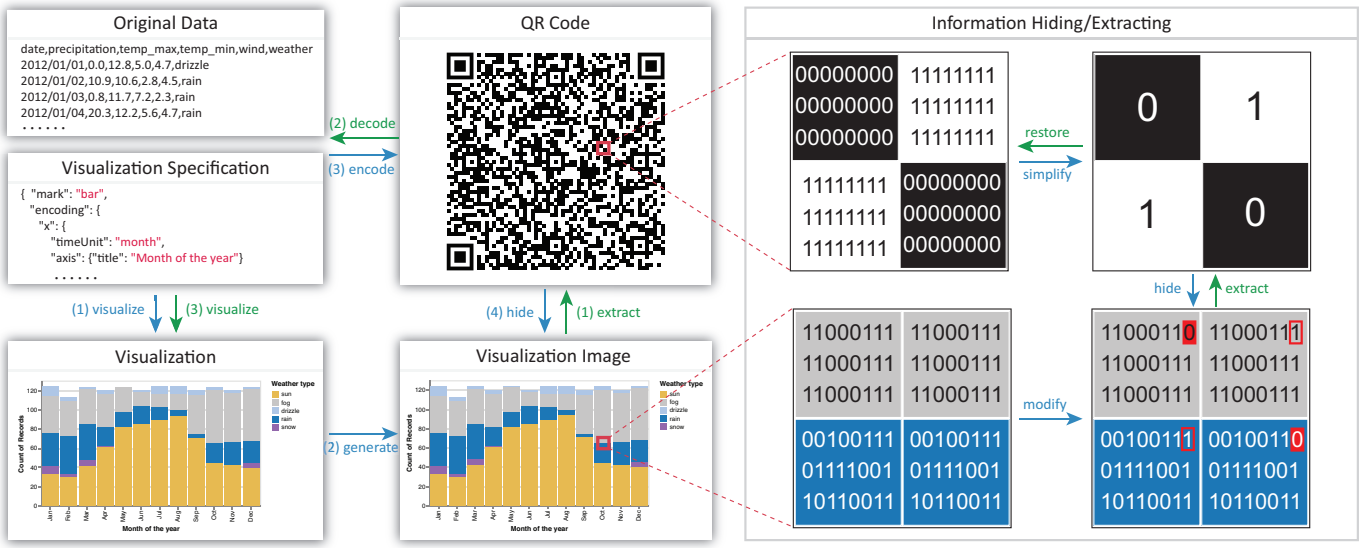

Fig. 7. The left part in the figure above shows the data hiding and extraction pipeline. The blue arrows in the figure above show the hiding procedures: (1) visualizing original data into interactive visualizations, (2) generating the visualization image, (3) encoding the original data and visualization specification into QR-code, and (4) hiding the QR-code into visualization images. The green arrows indicate the extracting procedures: (1) extracting QR-code from the visualization images, (2) decode the original data and visualization specification from QR-code, (3) visualizing the original data into interactive visualizations. The right part shows the detailed method using four matching pixels in the QR-code image and visualization image. Data hiding method firstly simplify the RGB value of each pixel in QR-code image into one bit, and then hide the each bit through modifying LSB in visualization image. The bits with red background is the modified bits. Data extracting method extracts the LSB from the visualization image and then restore to the pixel values of QR-code image.

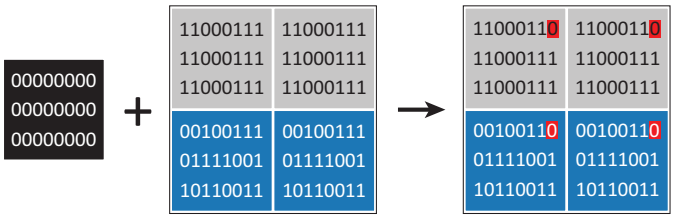

Fig. 8. The top part of the figure above shows that a QR-code can encode 524 alphanumeric characters. The bottom part are the damaged or dirty QR-code.

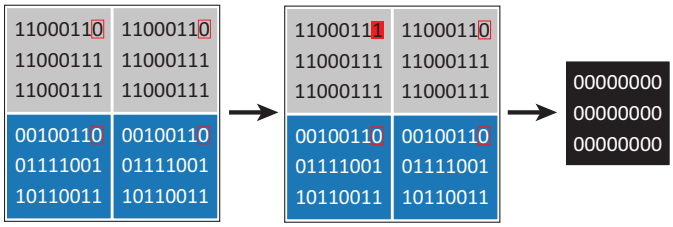

Fig. 9. The top part of the figure above shows that a QR-code can encode 524 alphanumeric characters. The bottom part are the damaged or dirty QR-code.

3.4 Extracting Method

which are divided into two parts, embedding data into visualization images and extracting the embedded data.

Our method integrates the target visualization image with original data through information hiding techniques with QR-Code. The pipeline as shown in Figure 1 describes the overview of our method. Compared with the traditional visualization process, there are a few extra procedures and intermediate results during the process from original data to integrated visualization. The followings is the details of the procedures and we will divide the procedures into two parts, data integrating and data extracting.

Data Integrating:

- ***Visualize***: It is the conventional procedure of data visualization. The input is the original data and output is the direct visualization, which also serves as the host image of QR-Code.
- ***Original Visualization***: Original visualizations only encode the data using visual elements without integrating original data.
- ***Encode***: It is the procedure that transforms the original data to QR-Code. The size and amount of QR-Code are determined by the data size and the host image size. The encoded data will be further explained in Section 4.1.
- ***Integrate***: It is to integrate QR-Code with the original visualization to get the integrated results. Hiding process has two inputs, one is the original visualization image, and the other is the QR-Code. The output is the integrated visualization, with hidden data which are imperceptible to users.
- ***Integrated Visualization***: It is the data hiding result which integrates visualization image with data. Also, it is spread from visualization authors to people in our application scenarios.

Data Extracting:

- ***Extract***: It is to retrieve the hidden QR-Code from the integrated visualization. It is the reverse process of ***data hiding***.
- ***Decode***: Decoding the hidden QR-Code could obtain the original data. This is the inverse process of ***data encoding***.

4 Technical Details

4.1 Data Extraction

5 Evaluation

In this section, we mainly evaluate the data hiding capability of our method. One of the application scenarios of our method is the visualization images online, so we collected 100 visualization images from

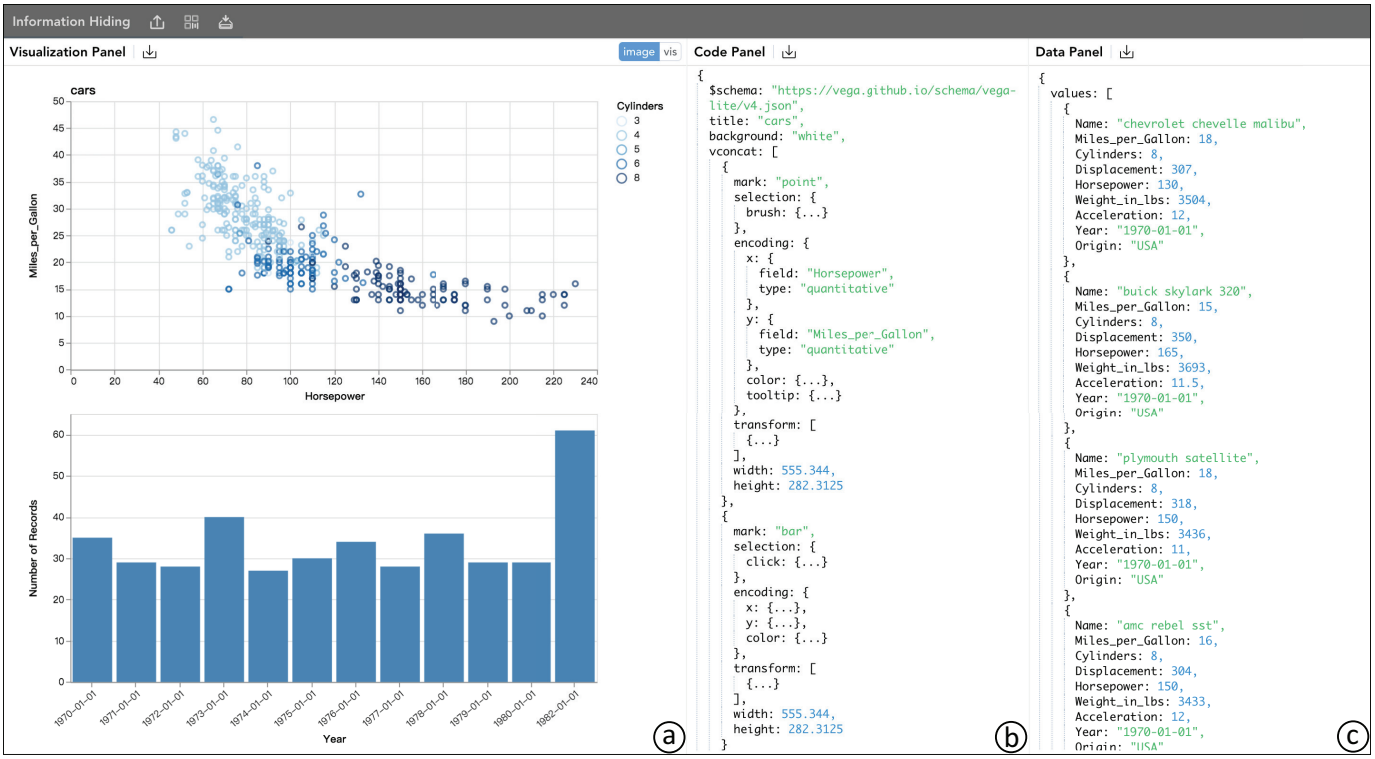

Fig. 10. The interface of the prototype system. (a) Visualization/Visualization image panel. (b) Visualization specification panel. (c) Original data panel.

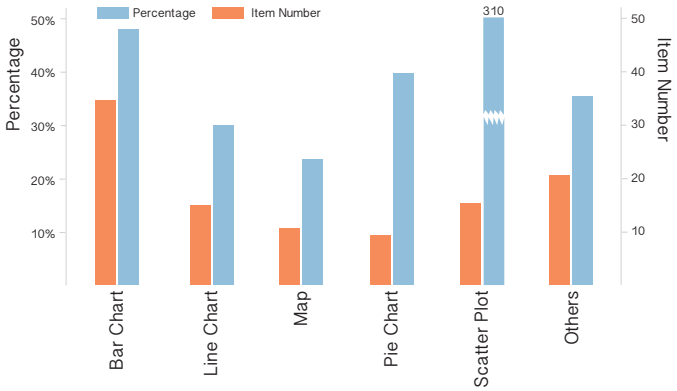

Fig. 11. The statistical results for DataNews visualization images. This chart shows the percentage of different visualization categories and the average number of data items.

DataNews website automatically. For each visualization image, we retrieve the underlying data manually. The statistical results of these images are shown in Figure 7. From these results, we could know that Bar Chart is most common in DataNews visualizations and most of the item numbers of the visualizations are in the range of 20 - 50 except scatter plot, which usually has a dense layout.

From the collected visualization images, we found that the minimum size of images is $550px \times 688px$, the average size is $794px \times 899px$. The amount of data that can be stored in the QR code symbol depends on the data type (mode, or input character set), version (1,$\cdots$, 40, indicating the overall dimensions of the symbol), and error correction level. The maximum storage capacities occur for 40-L symbols (version 40, error correction level L) [**?**] and maximum storage capacities for four different characters(Numeric only, Alphanumeric, Binary/byte, Kanji/kana) are different. In most cases, original data is alphanumeric and its maximum storage is 4296 characters. According to the average size of collected visualization images, the size of the gradient domain matrix is $397px \times 899px$, so it can accommodate 8 QR-Codes, and the maximum size is 34368 characters. In general, the original data could be integrated with the visualization images directly, so it validates the effectiveness of our method in current scenarios.

6 Results

As shown in Figure 6 and Figure 9, we have applied our method to hide data of different visualizations to get the integrated results and extract the hidden data. For some traditional visualizations, our restoration system supports to restore the visualization and make further exploration for them. The results for this part are shown in Figure 9(1)-(4).

Original Data Extraction. To test the effectiveness of integrating and extracting methods, we choose two kinds of visualizations used frequently in our daily life.

- **Bar Chart:** The first one is a bar chart, because it is the most common visualization for the masses. For the bar chart in Figure 6, the size of its original data is 223 characters and the size of the image is $600px \times 270px$, we hide the original data into the visualization using only one QR-Code and the extraction vector $= 106°$.

- **Box Plot:** The second one is a box plot, because it could not support people to retrieve the original data through the other extraction methods based on computer version techniques directly. The box plot has 323 characters in total. The image size is $200px \times 100px$, we hide two QR-Codes into visualization at the same time by "higher-order" image hiding algorithm and the extraction vectors $= 169°, 146°, 157°$.

Restoration and Further Exploration. In this case, we test restoration and further explorations on the bubble charts. Figure 9 shows the developed visualization restoration system and (1)-(4) show the

simulated exploration processes of several users. The restoration for its color encodings is realized by the hidden visualization descriptions.

The bubble chart counts the number of words in a passage, the color encodes the categories of different words and the size of circle encodes the frequency of the words. The visualization authors integrate the images with original data to get the integrated visualization. The people who got the integrated results could retrieve the underlying data and restore the visualization. Brushing the bubbles chart could explore the word count distributions. When the users have some significant findings, users may commit their findings which will be recorded in the exploration tree. Integrated visualizations can be shared between different users and their findings are also transmitted to others at the same time.

This paper introduces data hiding techniques to visualization and integrates visualization with original data through hiding QR-Code into images, however, our method mainly focuses on keeping the perception of original visualization through the imperceptibility characteristic of information hiding techniques rather than the data privacy.

While our extraction method can guarantee nearly 100% accuracy, it still has some limitations. This statement is founded only under the prerequisites that the visualization images are not damaged or compressed, otherwise the extraction accuracy will degrade significantly. Therefore our method has high requirements for the quality of visualization images.

Although the usage scenario in this paper is quite simple, we believe our method still offers potential to help people's further explorations and even the visualization spreading between masses, and we also think there are several open directions for future work.

1. Combining with Automatic Visualization Construction. Our method integrates and extracts data for all categories of visualizations. However, the visualization restoration system only supports some conventional visualizations currently, and in the future we will generalize the restoration part to more categories, and recommend the visualization automatically according to the data attributes.

2. Physically transmitted integrated visualization. In our method, the visualizations have to be delivered digitally, and can not be transformed physically using cameras or printers to capture the information, because it will introduce color distortion and compression. For the next step, we want to get another integrated visualization, which provides users overview of the visualization and support to be scanned using cameras to deliver the information physically. We believe it will improve the visualization spreading between masses greatly.

7 Conclusion

We propose data integrated visualization to provide users with the visualization and original data together. Our method enable viusalization authors to hide data into target visualization images through QR Codes to get the integrated visualizations. The hidden data is imperceptible to users, so the integrated results can keep users' perception of the original visualizations. Compared with other approaches, extracting data from integrated visualizations we proposed is more general, accurate and efficient. Furthermore, the visualization restoration prototype implemented based on this method supports users to restore the visualizations and make further explorations.

References

[1] J. Bertin. *Semiology of Graphics - Diagrams, Networks, Maps*. ESRI, 2010.

[2] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1121–1128, 2009.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[4] Z. Chen, Y. Wang, Q. Wang, Y. Wang, and H. Qu. Towards automated infographic design: Deep learning-based auto-extraction of extensible timeline. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):917–926, 2020.

[5] Q. Code. *https://www.qrcode.com/en/*.

[6] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker. *Digital watermarking and steganography*. Morgan kaufmann, 2007.

[7] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2008.

[8] J. Gao, Y. Zhou, and K. E. Barner. View: Visual information extraction widget for improving chart images accessibility. In *19th IEEE International Conference on Image Processing, ICIP 2012, Lake Buena Vista, Orlando, FL, USA, September 30 - October 3, 2012*, pp. 2865–2868, 2012. doi: 10.1109/ICIP.2012.6467497

[9] A. Gross, S. Schirm, and M. Scholz. Ycasd–a tool for capturing and scaling data from graphical representations. *BMC bioinformatics*, 15(1):219, 2014.

[10] L. Hao, J. Feng, and B. Zhou. Gradient domain binary image hiding using color difference metric. In *SIGGRAPH Asia 2015 Technical Briefs, Kobe, Japan*, pp. 4:1–4:4, 2015. doi: 10.1145/2820903.2820919

[11] J. Harper and M. Agrawala. Deconstructing and restyling D3 visualizations. In *The 27th Annual ACM Symposium on User Interface Software and Technology, UIST '14, Honolulu, HI, USA, October 5-8, 2014*, pp. 253–262, 2014. doi: 10.1145/2642918.2647411

[12] J. Harper and M. Agrawala. Converting basic D3 charts into reusable style templates. *IEEE Trans. Vis. Comput. Graph.*, 24(3):1274–1286, 2018. doi: 10.1109/TVCG.2017.2659744

[13] A. Hota and J. Huang. Embedding meta information into visualizations. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019. doi: 10.1109/TVCG.2019.2916098

[14] W. Huang, R. Liu, and C. L. Tan. Extraction of vectorized graphical information from scientific chart images. In *Proceedings of International Conference on Document Analysis and Recognition (ICDAR)*, pp. 521–525, 2007.

[15] W. Huang and C. L. Tan. A system for understanding imaged infographics and its applications. In *Proceedings of the 2007 ACM Symposium on Document Engineering, Winnipeg, Manitoba, Canada, August 28-31, 2007*, pp. 9–18, 2007. doi: 10.1145/1284420.1284427

[16] W. Huang, C. L. Tan, and W. K. Leow. Model-based chart image recognition. In *Proceedings of Graphics Recognition, Recent Advances and Perspectives (GREC)*, pp. 87–99, 2003.

[17] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, and J. Seo. Chartsense: Interactive data extraction from chart images. In *Proc. ACM Conf. Human Factors in Computing Systems (CHI)*, pp. 6706–6717, 2017.

[18] S. Katzenbeisser and F. A. Petitcolas. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, Inc., USA, 1st ed., 2000.

[19] G. G. Méndez, M. A. Nacenta, and S. Vandenheste. ivolver: Interactive visual language for visualization extraction and reconstruction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*, pp. 4073–4085, 2016. doi: 10.1145/2858036.2858435

[20] Min Lu, Jie Liang, Yu Zhang, Guozheng Li, Siming Chen, Zongru Li, and X. Yuan. Interaction+: Interaction enhancement for web-based visualizations. In *Proc. IEEE Pacific Visualization Symposium (PacificVis)*, pp. 61–70, 2017.

[21] J. Poco and J. Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. *Computer Graphics Forum*, 36(3):353–363, 2017.

[22] J. Poco, A. Mayhua, and J. Heer. Extracting and retargeting color mappings from bitmap images of visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):637–646, 2018.

[23] N. Provos and P. Honeyman. Hide and seek: An introduction to steganography. *IEEE security & privacy*, 1(3):32–44, 2003.

[24] A. Rohatgi. Webplotdigitizer, https://automeris.io/webplotdigitizer/, 2015.

[25] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. *Computer Graphics Forum*, 33(3):351–360, 2014.

[26] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017.

[27] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. Revision: Automated classification, analysis and redesign of chart images. In *Proc. ACM Conf. Symposium on User Interface Software and Technology (UIST)*, pp. 393–402, 2011.

[28] F. Y. Shih. *Digital watermarking and steganography: fundamentals and techniques*. CRC press, 2017.

[29] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Trans. Vis. Comput. Graph.*, 8(1):52–65, 2002. doi: 10.1109/2945.981851

[30] B. Tummers. Datathief iii, https://datathief.org/. 2006.
[31] S. N. P. Vitaladevuni, B. Siddiquie, J. Golbeck, and L. S. Davis. Classifying computer generated charts. In *International Workshop on Content-Based Multimedia Indexing, CBMI '07, Bordeaux, France, June 25-27, 2007*, pp. 85–92, 2007. doi: 10.1109/CBMI.2007.385396
[32] Y. P. Zhou and C. L. Tan. Hough technique for bar charts detection and recognition in document images. In *Proceedings of the International Conference on Image Processing (ICIP)*, pp. 605–608, 2000.