**Gallery D.C.: Auto-created GUI Component Gallery for Design Search and Knowledge Discovery**

**ABSTRACT**

GUI design is an integral part of software development. The process of designing a mobile application typically starts with the ideation and inspiration search from existing designs. However, existing information-retrieval based, and database-query based methods cannot efficiently gain inspirations in three requirements: design practicality, design granularity and design knowledge discovery. In this paper we propose a web application, called Gallery D.C., that aims to facilitate the process of user interface design through real world GUI component search. Gallery D.C. indexes GUI component designs using reverse engineering and deep learning based computer vision techniques on millions of real world applications. To perform an advanced design search and knowledge discovery, our approach extracts information about size, color, component type, and text information to help designers explore multi-faceted design space and distill higher-order of design knowledge. Gallery D.C. is well received via an informal evaluation with 7 professional designers.

Web Link: http://mui-collection.herokuapp.com/
Demo Video Link: https://youtu.be/zVmsz_wY5OQ.

**CCS CONCEPTS**

- Software and its engineering:
- \[\text{GUI design, multi-faceted design search, object detection}\]

**KEYWORDS**

GUI design, multi-faceted design search, object detection

**ACM Reference Format:**

Sidong Feng, Chunyang Chen, and Zhenchang Xing. 2022. Gallery D.C.: Auto-created GUI Component Gallery for Design Search and Knowledge Discovery. In *44th International Conference on Software Engineering Companion* (ICSE ’22 Companion), May 21–29, 2022, Pittsburgh, PA, USA. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3510454.3516873

**1 INTRODUCTION**

Graphical User Interface (GUI) is ubiquitous in almost all modern desktop software, mobile applications and online websites. It provides a visual bridge between a software application and end users through which they can interact with each other. A good GUI design makes an application easy, practical and efficient to use, which significantly affects the success of the application and the loyalty of its users [10, 12, 13, 18]. To design a good GUI, designers usually start with the ideation and inspiration search based on the initial user needs and software requirements [4].

Despite the enormous amount of GUI design search engines existed, it is still difficult for designers to efficiently get inspiration and understand design knowledge. Three requirements behind this phenomenon: design practicality, design granularity and design knowledge discovery. First, designers want to see the practical use of GUI designs in real applications, not just artworks, as GUI designs in artwork and real application have substantial differences. Second, designers want to see both the overall design and the detailed design of the GUI components. Although there are some GUI component design kits, but they do not provide the GUI context in which certain components can be practically composed. Third, designers want advanced GUI design search capabilities and knowledge discovery support.

To address this, we design and implement Gallery D.C.¹, a gallery of 11 types of GUI design components (e.g., button, image button, check box, radio button, switch, toggle button, progress bar, rating bar, seek bar, spinner, and chronometer) that harness GUI designs crawled from millions of real-world applications using deep learning based reverse-engineering and computer vision techniques. Gallery D.C. assists designers and developers to (1) access 68k practical GUI designs, 469k app-introduction GUI designs and 32k GUI component designs of 130k Android apps (2) enable multi-faceted search (e.g., component types, size, color, application category and/or displayed text) on 11 types component design (3) understand advanced design knowledge discovery through visualising design demographics (distributions of component type usage, primary colors, and height/width) and design comparison features.

Gallery D.C. represents a significant departure from and improvement over existing works and websites. For example, Dribbble [1] only shares GUI design artworks to exchange creativity and aesthetics. In contrast, our work exploits real world design resources, e.g., screenshots of real apps and app introduction in the application market. There are many related works on GUI design search [6–8, 14] but rarely related to GUI component search. The most closely related approach to our tool is Guigle [3] that utilizes automated information retrieval techniques and indexes metadata to return relevant GUI designs in relation to a query. While the Guigle approach represents a promising technique for helping developers to search GUI designs in granularity, it does little to help designers get inspiration on components as it returns GUI designs

**Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.**

ICSE ’22 Companion, May 21–29, 2022, Pittsburgh, PA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9223-5/22/05...
https://doi.org/10.1145/3510454.3516873

---

¹Gallery D.C. is an abbreviation for Gallery of Design Components
Figure 1: The overview of GUI component dataset collection for Gallery D.C.

Figure 1 provides an overview of dataset collection for Gallery D.C., including two main steps: app GUI screenshot collection and GUI component wirification. More details can be seen in our previous work [5].

2 DATASET COLLECTION

2.1 App GUI Screenshot Collection

We collected two kinds of app GUI screenshots, annotated runtime app screenshot and app introduction screenshot.

**Annotated Runtime App Screenshots:** We downloaded Android apps (e.g., APKs) from Google Play and dumped app metadata including app category, download times, etc. Based on the Android application GUI test framework, we developed an automated GUI exploration method which can automatically execute a mobile application and explore the GUIs of apps by simulating user interaction. During such automated GUI explorations, we used Android UI Automator [2] to dump runtime GUI screenshots and its corresponding GUI component metadata (e.g., component type, size and position). We further leveraged Rico dataset [11] to extend our dataset. This dataset contains 68k screenshots from 4k apps.

**App Introduction Screenshots:** The app introduction screenshot illustrates the most important GUIs of an app (usually the one with the best design), therefore, we were motivated to exploit this undiscovered GUIs gold mine to extend our gallery. We built a web crawler based on the breadth-first search strategy [15], e.g., collecting a queue of URLs from a seed list, and putting the queue as the seed in the second stage while iterating. Apart from the app introduction designs, our crawler also dumped its corresponding app metadata. This dataset contains 469k app introduction GUI screenshots of 126k Android apps.

2.2 GUI Component Decomposition

It is straightforward to decompose components from annotated runtime app screenshot which contains components metadata, however, the app introduction screenshots are purely images without any hierarchy information. Therefore, we proposed an approach to decompose GUI components from app introduction screenshots.

**Screenshot Augmentation:** The app introduction screenshots are more like posters which contain not only the app GUI, but also many other information such as app name, short descriptions. To reduce such difference, we first applied a GUI-screenshot specific image augmentation method (randomly adjust the size and position) to simulate the annotated runtime screenshots as the app introduction screenshots.

**Object Detection:** We then used the simulated annotated runtime screenshots to train a GUI component detection model using Faster RCNN [16]. The model used convolutional neural network (CNN) to extract image features and region proposal network (RPN) to generate Region of Interest (ROI). At the end, the model applied an object detection model to detect and extract different components. Thus, our trained model can automatically decompose the
app introduction screenshots into a set of GUI components and determine the component type, position, width and height.

3 TOOL IMPLEMENTATION AND USAGE

Gallery D.C. is a web app, which provides several convenient techniques for users to search, analyze, compare the GUI designs to gain inspirations.

Multi-faceted Search (Figure 2a): We allow the users to search for component class, category, height, and width, and these attributes can be easily indexed by metadata. Color design is an important aspect of GUI design. In order to perform search for color, we augmented primary color of GUI component by using HSV colorspace detection. Moreover, text search query allows the users to find all components of a text and visualize their design styles. To achieve this, we used Optical Character Recognition (OCR) technique to extract the text information on the GUI component.

Design Demographics (Figure 2b): To help designers understand the overall design landscape at an aggregation level, we dynamically compute design demographics for the multi-faceted search results, so that the designers can obtain a quick overview of the key characteristics of the search results. The system computes four types of design demographics: the component color (computed by HSV colorspace), the component height and width (obtained from object detection), the number of components of each component type (determined by object detection), the number of component of each application category (obtained from metadata). The system adopts Google Analytics Chart api to visualize the distribution in piechart (component class and color), scatter plot (component height and width), and bar chart (app category).

Component Information (Figure 2c): Once the user finds an attractive component, he can click on it to view more detailed information, such as the whole GUI (component is highlighted with red bounding box), app package, developers, etc. We provide users with some similar components to extend their design inspirations. The selection of similar components is calculated by the correlation of app, developer, color, text, etc. We also support the users to put the component into their design notebook for future reference, using "Share" button.

Design Comparison (Figure 2d): Design comparison allows designers to see the brand commonalities and variations of different companies. We borrowed the idea of comparison shopping to support design comparison. The gallery system allows designer to select different companies to compare their GUI component designs. Note that only high reputation/ranking companies are displayed (34 companies). As a company has many apps, and an app has many components, we only displayed the top six components from app with high downloads and high reputation. If the company does not have a particular type of components, the system reports "None" for this component type. The system also computes the color distribution of components from the company. Finally, it generates a comparison table for designers to visually compare GUI components side by side.

User Experience (UX): To build a scalable website, we adopted Node.js (Express as the framework) as it is asynchronous and event-driven. Since we had a large number of database, the efficiency of our website is crucial. We applied two methods. First, for the database side, we adopted MongoDB as our database program, because on the one hand, it is well integrated with Node.js; on the other hand, it enables more efficient large-scale processing of data. Second, we used the lazy loading strategy which is commonly used in image search engines. It showed some initial search results on the visible screen and loaded more search results when users scroll down the search results. The core technique of this strategy is an event listener which listens to the events in the browser such as scrolling, resizing, etc. After we recognized a change, we determined the space in the browser and calculated the number of images that should become visible on the screen using window height, image’s top offset, etc. Finally, we triggered them to load accordingly.

To keep the style of web pages consistent, we adopted the most popular CSS framework, Bootstrap. For the search page, to optimize the use of space when displaying components, we adopted a widely used layout, masonry layout. Instead of the common vertical masonry, we decided to set masonry in horizontal as most of the components are flat and wide.

4 EVALUATION

The goal of this section is to evaluate our platform Gallery D.C. in terms of (i) its performance in decomposing GUI components, and (ii) the usefulness of our Gallery D.C. in a real design environment. We elaborate more details and illustrations in our previous work [5].

4.1 Performance of Decomposition

Among all 68,702 annotated runtime GUI screenshots, we leveraged 90% to train our object detection model. We empirically set
We surveyed 7 professional designers to assess the usefulness of Gallery D.C. The designers responded positively to the use of the website. We collected and analysed their general feedback on how our website can be used for their own application design tasks.

**Usage Scenario 1: Design Granularity.** One designer works on designing an application for game casual. One of her major job is to decide the button style of a game application. She praised that our gallery is particularly useful for providing inspirations to her. Since the main target audience for her app is young girls, she was looking for buttons that fit young girl’s style. One button of pink color with reflective bubble was selected by her as this kind of design is deemed to attract young girls. To remind her of this design inspiration, she put the design in her design notebook for reference. She also found another button with ranking-list icon interesting. It inspired her not only the button semantic, but also reminded her to add a ranking mechanism into the game. In addition to the button design itself, she also learned where the button is composed in the real-world app and embeds into her own app.

**Usage Scenario 2: Design Practicality** One designer is creating a GUI component design system, which is a set of standards for designing and coding with components. This GUI component design system can enhance the visual and interactive consistency of different products within the company to provide a better and familiar user experience. In order to understand the design style of different companies, he adopted the comparison function provided in our Gallery D.C. He looked at two large companies (Google and Microsoft) for a detailed comparison. The comparison table presented representative GUI components of each company side by side. By this intuitive visual comparison, he clearly discovered that Microsoft prefers using right-angle rectangle components, while Google prefers using white, black and gray color as the dominant color and furthermore embedding the shadowing effects to the components. With those features displayed, he further thought about how to distinguish his design system from the current ones.

**Usage Scenario 3: Design Knowledge Discovery** One designer also provided another example of using our website to help the design of a social application. She searched the GUI components in social application category in Gallery D.C. On the one hand, lots of component designs were retrieved for her to gain inspirations. On the other hand, four demographics were reported to help her understand design knowledge. She observed that most buttons within the social-media apps are rather flat and wide as compared with other categories. After reviewing her experience in using social apps and searching the web, she further verified that this is a common phenomenon in social apps, because wide and large buttons can easily attract the attention of the audience, especially for the widespread young users in social apps.

Gallery D.C. equips with the summarization of GUI component designs is a powerful approach to understand the design trend and practices.

**5 CONCLUSION** In this paper, we present Gallery D.C., a novel GUI component design gallery system that can satisfy the designers’ information need for design practicality, design granularity, and advanced design search and knowledge discovery. We exploit a gold mine of app introduction screenshots in the app market to further extend our component gallery, using deep learning model and computer vision techniques. Finally, Gallery D.C. receives positive feedback from 7 professional designers who see its potential as a useful tool for their real design projects.

In the future, we will further remove some low-quality data within our gallery as the designers found that some retrieving results do not look professional, hence no reference value. Furthermore, several designers wonder whether our gallery could extend to other platforms, such as iOS, web.

**REFERENCES**

[1] 2021. Dribbble. https://dribbble.com/.
[2] 2021. UI Automator. https://developer.android.com/training/testing/ui-automator.
[3]Carlos Bernal-Cárdenas, Kevin Moran, Michele Tufano, Zichang Liu, Linyong Nan, Zhehan Shi, and Denys Poshyvanyuk. 2019. Gauge: a GUI search engine for Android apps. In Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings. IEEE Press, 71–74.
[4] Chunyang Chen, Sidong Feng, Zhengyang Liu, Zhenchang Xing, and Shengdong Zhao. 2020. From lost to found. Discover missing ui design semantics through recovering missing tags. Proceedings of the ACM on Human-Computer Interaction 4, CSCW2 (2020), 1–22.
[5] Chunyang Chen, Sidong Feng, Zhengyang Liu, Linna Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery DC: Design Search and Knowledge Discovery through Auto-created GUI Component Gallery. Proceedings of the ACM on Human-Computer Interaction 3, CSCW (2019), 180.
[6] Chunyang Chen, Ting Su, Gauzhu Meng, Zhenchang Xing, and Yang Liu. 2018. From ui design image to gui skeleton: a neural machine translator to bootstrap mobile gui implementation. In Proceedings of the 40th International Conference on Software Engineering. ACM, 665–676.
[7] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xin Xia, Liming Zhu, John Grundy, and Jinshui Wang. 2020. Wireframe-based UI design search through image autoencoder. ACM Transactions on Software Engineering and Methodology (TOSEM) 29, 3 (2020), 1–31.
[8] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xiwei Xu, Liming Zhut, Guoqiang Li, and Jinshui Wang. 2020. Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning. In 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). IEEE, 322–334.
[9] Jieshan Chen, Mulong Xie, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, and Guoqiang Li. 2020. Object detection for graphical user interface: old fashioned or deep learning or a combination? In proceedings of the 28th ACM joint meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 1202–1214.

[10] Qiuyuan Chen, Chunyang Chen, Safwat Hassan, Zhengchang Xing, Xin Xia, and Ahmed F. Hassan. 2021. How should I improve the UI of my app? a study of user reviews of popular apps in the Google Play. ACM Transactions on Software Engineering and Methodology (TOSEM) 30, 3 (2021), 1–38.

[11] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology. ACM. 845–854.

[12] Sidong Feng, Suyu Ma, Jinzhong Yu, Chunyang Chen, Tingting Zhou, and Yankun Zhen. 2021. Auto-icon: An automated code generation tool for icon designs assisting in UI development. In 26th International Conference on Intelligent User Interfaces. 59–69.

[13] Bernard J Jansen. 1998. The graphical user interface. ACM SIGCHI Bulletin 30, 2 (1998), 22–26.

[14] Zhe Liu, Chunyang Chen, Junjie Wang, Yuekai Huang, Jun Hu, and Qing Wang. 2020. Owl eyes: Spotting UI display issues via visual understanding. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE. 398–409.

[15] Marc Najork and Janet L Wiener. 2001. Breadth-first crawling yields high-quality pages. In Proceedings of the 10th international conference on World Wide Web. ACM, 114–118.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems. 91–99.

[17] Mulong Xie, Sidong Feng, Zhenchang Xing, Jieshan Chen, and Chunyang Chen. 2020. UED: A hybrid tool for GUI element detection. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 1655–1659.

[18] Tianming Zhao, Chunyang Chen, Yuanning Liu, and Xiaodong Zhu. 2021. GUGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE. 748–760.