Snap n' Shop: Visual Search-Based Mobile Shopping Made a Breeze by Machine and Crowd Intelligence

Quanzeng You*, Jianbo Yuan*, Jiaqi Wang†, Philip Guo* and Jiebo Luo*
*Department of Computer Science, University of Rochester, Rochester, NY 14627
Email: {qyou, jyuan10, pg, jluo}@cs.rochester.edu
†Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627
Email: jwang71@hse.rochester.edu

***Abstract*—The increasing popularity of smartphones has significantly changed the way we live. Today's powerful mobile systems provide us with all kinds of convenient services. Thanks to the wide variety of available apps, it has never been so easy for people to shop, to navigate, and to communicate with others. However, for some tasks we can further improve the user experience by employing newly developed algorithms. In this work, we try to improve visual search based mobile shopping experience by using machine and crowd intelligence. In particular, our system enables precise object selection, which would lead to more accurate visual search results. We also use crowdsourcing to further extend the system's prowess. We conduct experiments on user interface design and retrieval performance, which validate the effectiveness and ease of use of the proposed system. Meanwhile, components in the system are quite modular, allowing the flexibility of adding or improving different modules of the whole system.**

I. Introduction

Today, people are getting used to acquiring information through mobile platforms due to the popularity of smartphones and accessibility of wireless networks. For instance, they use smartphones for shopping, navigation, communication and so on. Current mobile systems become increasingly powerful in terms of providing a wide variety of services. Most of the time, simple installation of several apps will meet people's most daily needs.

A large portion of many app's success can be attributed to the incorporation of more and more machine intelligence. Over the past couple decades, machine learning and computer vision communities have developed more and more robust features and algorithms for object recognition and detection. In particular, the recently developed deep learning enables robust and accurate feature learning, which in turn produces the state of the art performance on image classification, objects detection and attribute learning. Both the academia and industry have invested a huge amount of effort in building powerful neural networks. Indeed, many companies have deployed deep neural networks into their products to improve the overall performance of their current systems. For instance, in June 2013, Baidu launched its image search platform based on deep learning[1].

[1]http://www.wired.com/wiredenterprise/2013/06/baidu-virtual-search/

However, there are still many problems which cannot be solved effectively using today's intelligent machines alone [1]. These tasks may be easily solved by human brainpower. Amazon Mechanical Turk provides a platform to use crowd intelligence to solve problems. Indeed, more and more people start to participate in human-based computation, which makes it feasible to assign tasks to human beings. We focus on a visual mobile shopping system. To be more specific, we want to use visual information to find corresponding products in a precise and easy fashion. This is particular useful when people do not know how to describe the visual object in terms of text. Our goal is to exploit both machine and crowd intelligence to deliver more pertinent visual search results.

Indeed, there are several apps that provide the function of searching products by images. Several of them are quite popular and interesting, including Amazon Flow, Ebay Fashion, CamFind and TapTapSee. We are not aware of any research systems in academia. However, there are several issues with these apps: a) Some of them are only concerned about its own products, for instance Amazon Flow can only recognize objects which are sold in Amazon. b) Due to limited image libraries, some of them can only detect objects that belong to certain categories. For instance, Ebay Fashion can only return a collection of women dresses. c) None user-friendly interfaces often lead to confusions when users are trying to figure out how to use the apps. For example, Amazon Flow only shows a patch of feature points. Whenever there is no target items, the app just stops there without giving any hint of its status. d) Most importantly, all of these apps do not allow users to specify the potentially interested objects in the query image. This may lead to unexpected or unwanted retrieval results if the query image contains multiple objects or background clutters.

In this work, we propose a user-friendly system, Snap n' Shop, which can provide the user with superior performance in terms of both retrieval system performance and integrated mobile user experience. In particular, we focus on the following several aspects of the system and make contributions.

- We provide multiple ways to select the query image, including directly taking a snapshot, choosing an image

from existing albums and downloading images from the Internet.
- Our app allows users to select the *exact* object they are interested in, which can greatly reduce unwanted retrieval results by excluding unwanted objects.
- Our system provides the flexibility to retrieve both textual and visual information of the object.
- We make use of crowdsourcing to allow workers with pertinent knowledge to help when the automatic image search does not yield good results.

Therefore, our system can avoid the problems caused by a limited image library by taking advantage of the retrieval results from the largest image search providers, such as Baidu and Google.

II. Related Work

Recent development in mobile technology has changed the way how people are using multimedia for information retrieval. In particular, mobile platforms have brought both challenges and opportunities for both industry and academia to develop new techniques for better multimedia mining.

It has been several years since Google first released its newly designed search engine for desktop computers: search by images. Before that, Google released its product Google Goggles for mobile visual search. Since then, there have been several different apps that provide similar service of searching by images. In general, these apps can be categorized into three groups. In the first group, the service is simply a natural extension of their service on desktop computers. For instance, companies such as Google, Bing and Baidu have released their own mobile apps. Most of the time, they will integrate many of their existing services into their apps to provide similar experience for their mobile users. However, it seems they do not provide specifically designed UI features to improve better search experience. In the second group, E-commerce companies such as Amazon and Ebay, released apps enabling users to use images for searching products. However, the problem is that they can only search the items from their own inventories, which may result in total irrelevant results otherwise. For example, Ebay can only return pictures that belong to the clothing category. On the other hand, Amazon Flow requires the image to include highly distinguishable features like barcodes and texts. The last group is most encouraging. Startup companies such as Image Searcher, Inc. developed apps (CamFind and TapTapSee) that aim at providing better performance when searching by images. In terms of accuracy, they perform much better than other similar apps that belong to the first two groups. However, they have the same issue: lack of integrated user interface support.

At the same time, there are also several related works on mobile visual search system. Girod et al. [2][3] proposed a visual search system in 2011. They developed new features and data structures for better feature extraction and retrieval. Since they used the 3G wireless network, one of their objectives is to reduce transmission delay, which may not be a big issue today. Meanwhile, the performance of their system relied on the database they built, which may not be suitable for arbitrary visual searches. However, their work showed that one could benefit from assigning some computational tasks on the phone. Schroth et al. [4] developed an location recognition system for mobile platforms. They developed novel feature quantization to overcome the limited computing power of mobile platforms. The main task is to identify the location of one picture, which is different from our system. The work in [5] also proposed a mobile location search system. They tried to predict the best viewing angle to allow more successful queries. Shen et al. [6] proposed a framework to implement automatic object extraction. By employing the top-retrieved images to accurately localize the object in the query image, their proposed framework significantly improved the retrieval performance. Their work also suggested the importance of extracting the query objects for improving retrieval performance. Most of visual retrieval systems are designed for single object retrieval. To enable multiple object recognition, the work in [7] employed a bottom up search-based approach. In this way, graph cuts can be used to solve the multi-object recognition problem. Meanwhile, in [8], the authors described an interactive multi-modal visual search system for mobile platforms. They took advantages of the multi-modal feedback from the user to retrieve the most matched images.

However, all of these works relied on their own-built database for objects retrieval. For more general purpose image retrieval, one has to rely on more complete image databases. For example, the authors in [9] proposed the Stanford Mobile Search data set for mobile visual search applications. Even though the data set contains different categories such as CDs, books, and outdoor landmarks, it is still not rich enough for general purpose image retrieval. On the other hand, search engine companies like Google and Baidu have built the world's largest general purpose image retrieval systems. Billions of images have been collected and indexed in their systems. In addition, they have built systems that adopted the most advanced algorithms to retrieve most relevant images. In this work, we make no assumptions on the image data base. Instead, we rely on the retrieval results from existing general image retrieval systems and indirectly exploit their knowledge database. We focus on developing new features which can generate more informative query images for the existing knowledge database.

III. System Architecture

Most of the current mobile visual search systems consist of two main components: client and server. Most of the time, the server contains a database which includes different categories of objects to retrieve in the system. The server needs to both process the request from different mobile clients and finish the retrieval task based on the database. To achieve efficient retrieval performance, researchers have provided different features and data structures to improve the retrieval performance [10][11][12][13]. Fig. 1 shows the architecture of our system. It has three main components: client, server and knowledge warehouse. In our implementation, the client

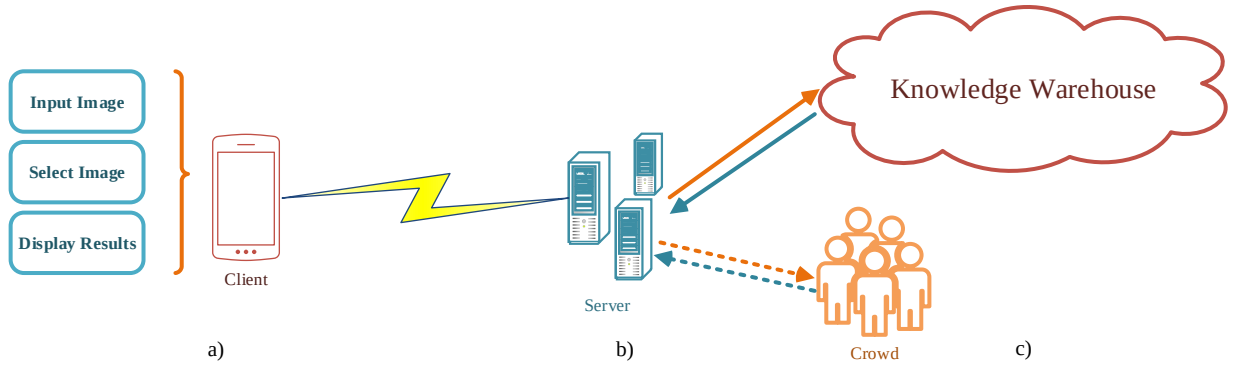


Fig. 1: Three components of our system: a) Client: we focus on designing a friendly user interface in smartphones to enable common gestures and easy selections on the image. b) Server: the server first accepts the request image from the client. It then sends out the request image to an online image search service. c) Knowledge Warehouse and Crowd Intelligence: the server sends out the request to the knowledge warehouse, which is capable of retrieving the most relevant information from the Web. If the warehouse fails to respond to our request or if there are no related results, the server will automatically resubmit the request to crowdsourcing marketplace.

is a mobile app running on the Android platform. We deploy a web server to receive requests from clients and the server will also be responsible for receiving the retrieval results from the knowledge warehouse. In this way, we transfer the retrieval task to the knowledge warehouse, and increase the overall throughput of our system. Next, we will discuss the three components in detail.

A. Client

For mobile platforms and apps, user experience mainly comes from the design of the user interface. The focus of the client is to develop a user friendly interface for the app. In addition, in order to improve effectiveness and efficiency of the proposed system, we also need to consider providing more effective input for our knowledge warehouse. In particular, we rely on computer vision techniques to help improve the user interface and the system's overall performance. To target on a specific object or region in the query image, our app enables the user to select and extract the interested objects from the image in a clean fashion. We implemented lazy snapping [14] to help users to select the objects from an image. Fig. 2 shows the main steps. First, the user chooses one image that contains the object of interest. Next, we run a superpixel segmentation algorithm on the image. We then use the superpixels as the graph nodes to build the graph for lazy snapping. Simple linear iterative clustering (SLIC) [15] is employed to extract all the superpixels from one image. More details and evaluation of the UI design will be discussed in the next section.

Today, most of the smartphones are configured with powerful CPUs and large memory. Therefore, we can implement some of the computational tasks in the client side. However, the bottleneck of today's smartphones is its battery life. To use the computational power more efficiently, we use the Java Native Interface[2] to implement both SLIC and Lazy Snapping, which enables us to precisely and cleanly select the object of interest from the query image.

B. Server

The server maintains the communication between the client and the knowledge warehouse as shown in Fig. 1. We use the Apache web server to receive HTTP requests from the clients. Since it can be easily configured with threadpools[3] to satisfy simultaneous multiple requests from different clients. For each received request, the server will automatically start a thread to handle the request. It queries the knowledge warehouse with the request image. We have no assumption on the architecture of the knowledge warehouse, thus different knowledge warehouses may have different result formats. The server has one independent component, result processor, to format the results. It is convenient to implement different result processors for different kinds of knowledge warehouses. Another main concern about the server is its reliability. It will severely affect user experience of the app in an undesirable way if the server fails to respond to the quest without any hints or responds too slowly to the request. We use Amazon EC2 to deploy our server. The main advantages with EC2 virtual machine are its reliability and the fact that we can choose to configure the server with the resources we require.

C. Knowledge Warehouse

Knowledge warehouse is a black-box component in our system. It is not important how to build a knowledge warehouse and we do not intend to build a new knowledge warehouse in this paper. We are mainly concerned about the programming interface provided by the knowledge warehouse. As long as the knowledge warehouse provides the interface to accept query images and return retrieved information, we can integrate this knowledge warehouse into our system.

[2]http://docs.oracle.com/javase/6/docs/technotes/guides/jni/

[3]https://commons.apache.org/dormant/threadpool/

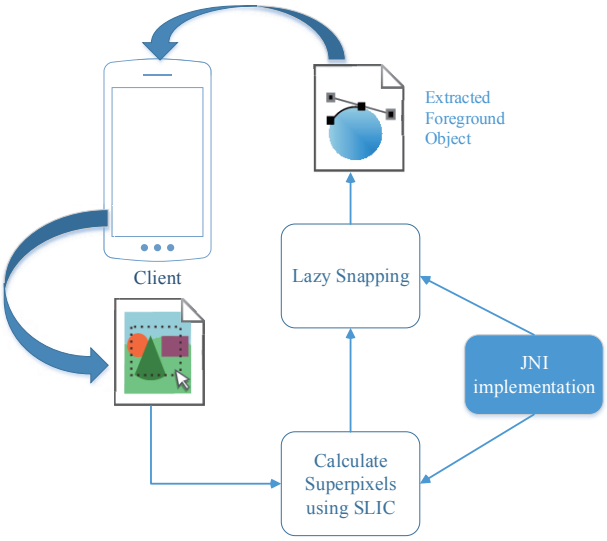


Fig. 2: Steps to extract the object of interest from one image.

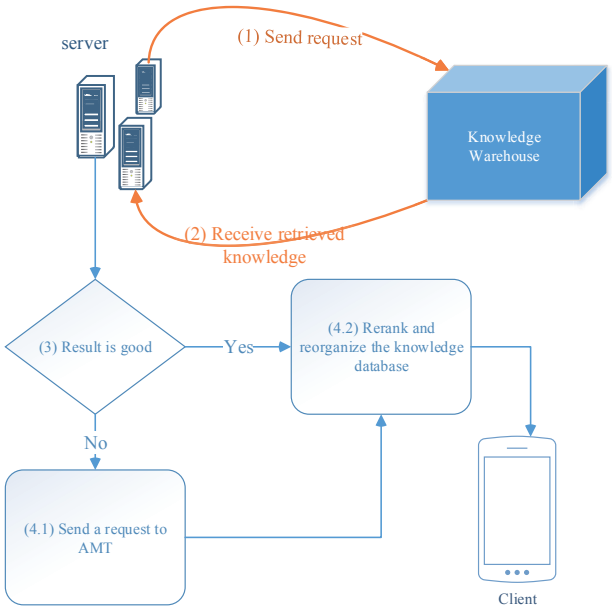


Fig. 3: Request process of the server.

In our implementation, we tried different versions of knowledge warehouses. In our initial attempt, we build our own database as a knowledge warehouse containing many different kinds of handbags. We implement our own data crawler to download the title, price, and text description of each item. In this way, we are able to build a database that contains details of items in many different categories. Meanwhile, for each item, we also download the images of that item under different views. After that, we build the index for each image using CEDD [16] features. Our experiments show that even for this simple handbag recognition task, the performance of the system is not always satisfactory. On one hand, the scale of our database is limited. Therefore, most of the time, the system could not find the item we are looking for and only return similar images. On the other hand, CEDD features are not rich enough to distinguish the slight differences between different handbags. Therefore, it may be problematic to use this feature for accurate image retrieval.

Given the above limitations, we turn to existing knowledge warehouses with more accurate and more comprehensive retrieval capabilities. In terms of accuracy and capacity, search engine companies own the world's largest knowledge warehouses that contain all different kinds of information. Moreover, Baidu image search[4] has relied on the state of the art technique, deep learning, for retrieving similar images of the query image. In our implementation, we chose to use Baidu image search as the knowledge warehouse for accomplishing the retrieval task. For each query image, the server will try to acquire results from Baidu Image. However, if Baidu fails to respond to our request or if there are no related results, the server will automatically resubmit the request to Amazon Mechanical Turk (AMT). Fig. 3 shows the main task for the server. To be more specific, if the retrieval result for one query is acceptable, the server will process the retrieved information and send it back to the mobile client. However, when the result is not acceptable or even there is no returned result, we create an AMT hit and ask people to manually solve this problem by using Amazon Mechanical Turk Command Line Tools[5]. For each hit, we request the worker to provide at least four or five kinds of tags including name, brand, category and possible link to this image. In this way, when the AMT workers finished the task, we can collect the results from the AMT server and return the results from the workers.

Even though we use Baidu image search for in this paper, as discussed above, our system can easily integrate with other knowledge warehouses. All we need to do is to implement the corresponding request processor and result processor according to their programming interface.

IV. UI Design Supported by Computer Vision

The main goal of our system is to retrieve knowledge for image objects. The main panel of the app provides different ways to choose the images. We offer three ways to choose the images that contain the object of interest: 1) Directly taking a snapshot. This is the most natural approach for people to find the information of one object. It is convenient for people to open the app and take a picture whenever they need to know detailed knowledge about one object as they see it. 2) Selecting the image from an image gallery. People may have taken many images on a trip of things they are interested in and later need to find out details on some of the objects from images in the image gallery. 3) Downloading images from a given URL. Users may notice some interesting images when browsing their Facebook pages, Twitter or some news articles from their smartphones. In this case, it is convenient to copy the URL of that image and paste it in our app. Next, the

[4]http://images.baidu.com/

[5]http://aws.amazon.com/developertools/Amazon-Mechanical-Turk/694

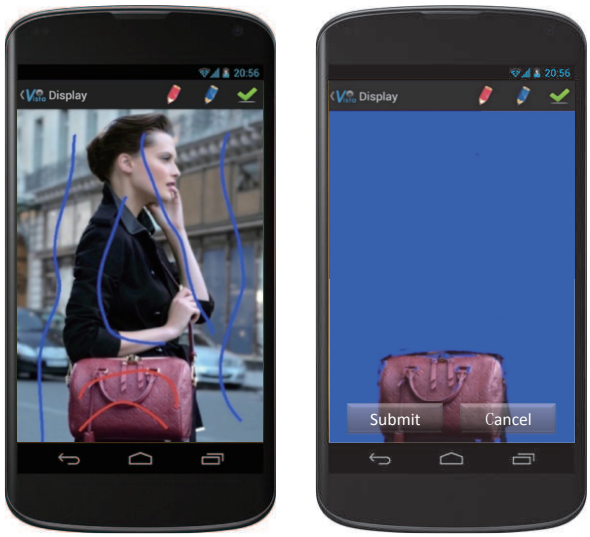
app will download the image and use that image as the query image automatically.

The design of organizing the three different approaches on the main panel allows the user to have a convenient and effective way to specify what they are looking for. In addition, we design the user interface for object selection along with different gesture support.

- **Scribbles**
  Our app provides gesture support for background and foreground object segmentation. In particular, we allow users to give hints on the two classes. We use different colors to represent the background and the foreground objects. Users can select two different colors to label areas for the background and foreground objects respectively to provide the supervision information. Moreover we allow multi-area labeling which is convenient for the users to select as many areas using as many scribbles as they wish.
- **Lazy Snapping**
  Based on the labeled areas, we use lazy snapping for extracting objects from the image. This is a semi-supervised way of extracting objects. The algorithm can incorporate the prior knowledge provided by users through scribbles to infer the foreground object. Users can provide knowledge of both the foreground and background objects. It is natural on smartphones to acquire such kind of supervised information through user-interface interaction due to the rich gestures on these mobile platforms. The objective for lazy snapping is to assign labels (foreground and background) for each pixel in one image with respect to minimizing a Gibbs energy $E(X)$. X is the label matrix for each pixel in the image. The Gibbs energy contains two components: likelihood energy and prior energy and is defined in Eqn.(1), where C refers to the set of pixel pairs that are connected.

$$E(X) = \sum_i E_1(x_i) + \alpha \sum_{i,j \in C} E_2(x_i, x_j) \qquad (1)$$

  As shown in Fig. 2, we first employ a superpixel algorithm to segment the whole images into different superpixels. After that we use the mean color value in each superpixel as the color for that superpixel to run the lazy snapping algorithm. Since the procedure is running on smartphones, the algorithm can be implemented as an iterative interaction between the app and the users. Each time, after the user select some background and foreground pixels though scribbles, the algorithm will try to find the object. If there is some imperfection in the extracted object, the user can give further hints for both the foreground and background objects. Our system can then return more accurate object extraction results. Fig. 4 shows one example of our app. We design three menu buttons, where the red pen and blue pen allow the users to select the foreground and background, respectively. The click on the checkmark button will call the lazy snapping algorithm to extract the targeted object. Fig. 4b is the extracted handbag from a test image. When users are satisfied with the extraction, they can submit the object to the server. Otherwise, they can press on the cancel button and continue to provide additional hints for both the background and foreground.



(a) BG and FG selection (b) Result of Lazy Snapping

Fig. 4: Users can select any area in the image as background (BG) and foreground (FG) with different colors. The right figure is the results in our mobile platform based on the scribbles in the left figure. Note that the scribble directions do not matter.

V. Retrieval Performance Study

In this section, we study the retrieval performance of requesting with the extracted objects from the image. By extracting the object from the image, we can obtain more robust retrieval results since we eliminate the noise background from the image. We conduct experiments to compare the performance of different systems by using the whole image and using only the extracted object from that image. In particular, we compare the retrieval results on two of the largest image search service providers, Google Image Search and Baidu Image Search.

Fig. 5 shows one retrieval example. The first column contains three different query images. The other two columns are the retrieval results from Google and Baidu, respectively. We have three different query images: the original image, the cropped object using bounding box and the cropped the object using lazy snapping. In general, we can observe that the retrieval results are most satisfactory when we preprocess the query image using lazy snapping. For the original image, Google prefers to return articles having the exact query image, while on the other hand Baidu tends to focus on the people in that image instead of the handbag. It is difficult to determine which one is better if we do not know what exactly the users are looking for. However, it seems that both of the retrieval results become worse when we query with the cropped handbag using a bounding box. In this case, both of them are trying

(a) Original Image (b) Google (c) Baidu

(d) Bounding Box (e) Google (f) Baidu

(g) Snapping (h) Google (i) Baidu

Fig. 5: Retrieval results from Google and Baidu image search respectively based on different query images of the object of interest. Note that our app is the only one that produces the exact match of the handbag.

to find out images that have similar colors with the query image. Besides, the images returned seem to be semantically different from the query images. However, if we further crop out the bag using lazy snapping, it seems helpful for both of the service providers. In the last row, we see that Baidu image search can find exactly the same handbag that we are looking for. Meanwhile, Google can retrieve objects containing similar categories and colors with the query image.

In general, it is difficult to know exactly what an individual user is looking for by the images provided. In particular, when there are multiple objects in the image, it becomes much more difficult for the system to determine the most desired object and to correctly return the most related information accordingly. However, the results in Fig. 5 suggest that we can exclude irrelevant retrieval results by providing more target-object centric images. Furthermore, the retrieval results from Baidu are more accurate than Google in most cases. Therefore, we use Baidu Image Search service as our knowledge warehouse in the following experiments.

VI. Analysis of User Experience and Performance

In our experiments, we compare the proposed Snap n' Shop (SNS) system with Amazon Flow, which is a well-known visual feature based retrieval system. Moreover, we also implement SNS Lite system, where we replace the lazy snapping component with a simple bounding-box based object selection component. The proposed systems described above have been evaluated in a controlled experiment. We evaluate the system mainly on two aspects: usability and user experience. In terms of usability we consider both retrieval result evaluation and user feedbacks. In terms of user experience, we evaluate user experience primarily on UI design, learnability, and overall rating comments. Moreover, we analyze the specific advantages and drawbacks of each system.

In the following subsections, we first describe our evaluation methodology and then report and discuss the results of usability and user experience evaluation.

A. Evaluation Methodology

In terms of usability, we conducted retrieval result evaluation by testing the proposed systems on a randomly chosen subset of ImageNet [17]. Moreover, in terms of both usability and user experience, we have conducted a controlled experiment with 10 participants (7 males, 3 females) from different academic backgrounds such as engineering, finance, and science. Each single-user session had a duration of 20 minutes for testing the three apps. Users are asked to test each system by querying randomly picked items under these three categories: *daily use, clothing & shoes, electronic products*. Evaluation process is completed by requiring a questionnaire containing 10 questions from our users, each of which covers one aspect of our evaluation tasks. Our questionnaire is presented in TABLE I.

1) Usability:

a) Test on Image Net: We test both of our proposed system and Amazon Flow on a subset of ImageNet. The subset contains 54 images and is randomly chosen from three categories including clothing & shoes, daily use (cosmetics, food) and electronic products. These images are used as queries and are fed into both systems to obtain retrieval results. In terms of performance evaluation, we assigned three well-trained judges to manually label each result. Manual labels are accepted if and only of all these three judges agree with each other.

b) User feedback: We evaluate user feedbacks on the retrieval results returned by all the platforms using collected answers from our questionnaire. Questions are shown in TABLE I.

2) User Experience: In terms of user experience, we evaluate our system by focusing on learnability and user satisfaction. Learnability is measured as how easy it is to use the app for the first time. User satisfaction is measured by overall rating of the app and UI design in terms of both aesthetical functionality perspectives and user feedbacks. We set up the experiments as discussed in Section VI-A1.

B. Results

In this section, we analyze the results the three apps using the experiments designed in Section VI-A.

1) Usability:

a) Retrieval Result Evaluation: TABLE II shows the overall retrieval performance tested on a randomly chosen subset of ImageNet. The HIT column shows the percentage of queries that receive the exact information about the query

TABLE I: User Interface Study Questions using the Likert scale.

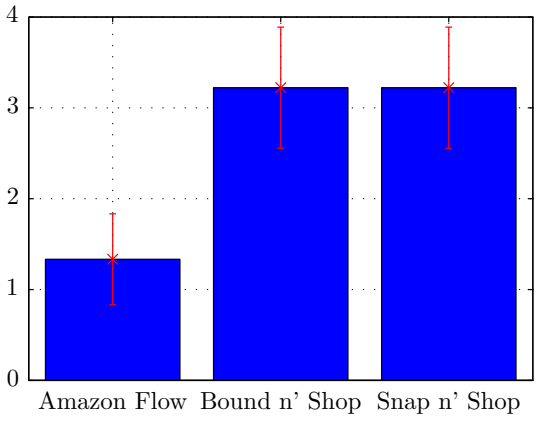
Usability	How are you satisfied with the retrieval results?(rating 1-5, 5 mean totally satisfied)
	Contains any features that others do not have? (yes/no)
	Do you like to use these new features? (rating 1-5, 5 mean loved it)
Learnability	Any instructions for you to learn how to use different functions? (yes/no)
	Do you find it is easy to use? (yes/no)
	Did you make any mistakes in the process of usage? (yes/no)
	How often do these mistakes happen? (rating 1-5, 5 means too many times)
User Satisfaction	Do you like the aesthetic design of this product? (rating 1-5, 5 means love it)
	Overall rating for this app (rating 1-5, 5 is the highest)

TABLE II: Retrieval performance tested on a subset of ImageNet

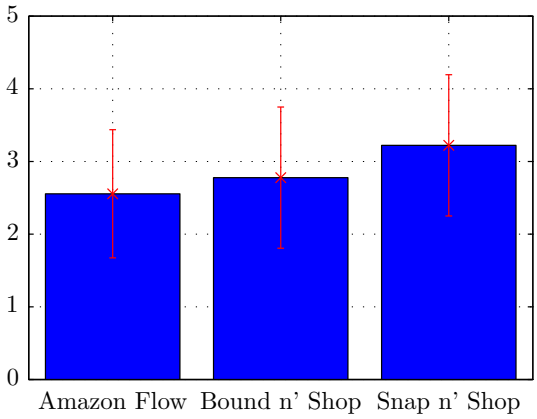
	HIT	Relevance	Success
Amazon Flow	0.148148	0.185195	0.296296
Snap n' Shop	0.351852	0.814815	1

image. The relevance column shows the percentage of queries that receive same category items. The success column is the percentage of queries which can receive responses from the tested app. Amazon Flow can only return results for very few queried samples. About 15% of them are the exact hits and the remaining returned results are actually detected based on texts in the query image. To be more specific, if one query image has textual information such as brands and titles, Amazon Flow can recognize the textual contents in nearly real time and use these detected features as queries. In our experiment, 6 out of 16 times it recognized the textual content and returned results based on these textual features. Among all the accepted queries, 50% of which are not what we are looking for or even not under the same category and 62.5% are labeled as relevant by our researchers. On the other hand, our system can provide results for each query image due to the nature of our knowledge warehouse. Overall we achieved about 81.5% relevant results and 35.2% are the exact hits. Note that both of our proposed systems are implemented with the same knowledge warehouse. Therefore the retrieval results performance for both systems are consistent with each other and we only show the result for SNS here. We can see that Amazon Flow can be easily distracted by the background. On the other hand, our results are more general and more robust against background distractions.

b) User Feedback: For usability study, we mainly focus on the overall retrieval results and each app's unique features. Fig. 6 shows the average user satisfaction scores of the the three apps. It seems that subjects prefer the retrieval result from Baidu. Even though we use the same knowledge warehouse (Baidu image search) for both SNS Lite and SNS, subjects still have higher satisfaction scores with the retrieval results from SNS compared with SNS Lite. This strongly validates our assumption that using the extracted exact object can improve retrieval performance. Meanwhile, different apps have different ways to capture the targeted objects. We denote these unique features as new features for each of them respectively. In Fig. 6b, we show the average scores for these new features. Compared with the way of using feature points for object detection (Amazon Flow), the subjects prefer to use the proposed interactive lazy snapping to extract the object. Analysis of the user study shows that people are used to using different gestures to interact with an app. Unfortunately, Amazon Flow does not allow these kinds



(a) Retrieval result



(b) New features

Fig. 6: Usability scores from user feedback in terms of retrieval results and new features respectively. Note that the confidence intervals are also indicated.

of interaction, making the users feel frustrated.

C. User Experience

User experience is another significant aspect when evaluating a mobile app. We evaluate our proposed system and Amazon Flow in terms of learnability and user satisfaction.

TABLE III: User experience ratings for Amazon Flow, SNS Lite and SNS. The first three questions are "yes/no" questions. Note that 8/10 refers to 8 out of 10 users. Note that higher ratings are better except for error proneness.

Questions	Amazon Flow	SNS Lite	SNS
Instructions	5/10	4/10	8/10
Easy to use	1/10	6/10	6/10
Error prone	10/10	0/10	5/10

Learnability.

To measure learnability, we focus on the following aspects: how easy it is to use the app to complete a task without any external instructions and how often, if any, errors occur when users are conducting normal operations on the systems. TABLE III shows statistical results collected from questions in the learnability category in TABLE I. Even through all three systems provide tutorials when the app is launched for the first time, Amazon Flow is reported as not easy to use and all of the participants reported to run into errors. On the other hand, SNS Lite and SNS are rated as easier to use with lower error occurrence. Due to its simple functionality, SNS Lite has the lowest error occurrence. However, the overall mistake ratings for SNS Lite and SNS are comparable and both of them are much lower than Amazon Flow.

User Satisfaction.

In this subsection, we evaluate our proposed systems and Amazon Flow in terms of focusing on user satisfaction including general UI design, overall rating and user feedbacks. According to our results, all of these three systems have distinctive functionality from each other. Amazon Flow is rated as the best in terms of aesthetical UI design, our users are mostly attracted by its real time feature points that are shown on the screen, which also results in a high average rating as a distinctive feature. However, in terms of functionality UI design Amazon Flow is rated relatively lower due to its non-interactive design that often leads to confusions as indicated in its low learnability. On the other hand, SNS is rated relatively high in terms of aesthetics design and distinctive feature.

VII. Conclusions

Mobile visual search is an challenging and interesting area of research and development. In this paper, we employ both machine and crowd intelligence to improve the performance of mobile visual search. Indeed, the user experience and the usage of machine and crowd intelligence can complement each other. The experimental results suggest that by integrating machine and crowd intelligence into a user friendly app, we are able to not only improve the retrieval results, but also provide better user experience. We hope our work will encourage the community to employ new techniques to improve user experience.

References

[1] L. Von Ahn, "Human computation," in *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*. IEEE, 2009, pp. 418–419.

[2] B. Girod, V. Chandrasekhar, R. Grzeszczuk, and Y. A. Reznik, "Mobile visual search: Architectures, technologies, and the emerging mpeg standard," *MultiMedia, IEEE*, vol. 18, no. 3, pp. 86–94, 2011.

[3] S. S. Tsai, D. Chen, V. Chandrasekhar, G. Takacs, N.-M. Cheung, R. Vedantham, R. Grzeszczuk, and B. Girod, "Mobile product recognition," in *Proceedings of the international conference on Multimedia*. ACM, 2010, pp. 1587–1590.

[4] G. Schroth, A. Al-Nuaimi, R. Huitl, F. Schweiger, and E. Steinbach, "Rapid image retrieval for mobile location recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2320–2323.

[5] R. Ji, F. X. Yu, T. Zhang, and S.-F. Chang, "Active query sensing: Suggesting the best query view for mobile visual search," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 8, no. 3s, p. 40, 2012.

[6] X. Shen, Z. Lin, J. Brandt, and Y. Wu, "Mobile product image search by automatic query object extraction," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 114–127.

[7] C.-C. Wu, Y.-H. Kuo, and W. Hsu, "Large-scale simultaneous multi-object recognition and localization via bottom up search-based approach," in *Proceedings of the 20th ACM International Conference on Multimedia*, ser. MM '12. New York, NY, USA: ACM, 2012, pp. 969–972. [Online]. Available: http://doi.acm.org/10.1145/2393347.2396359

[8] H. Li, Y. Wang, T. Mei, J. Wang, and S. Li, "Interactive multimodal visual search on mobile device," *Multimedia, IEEE Transactions on*, vol. 15, no. 3, pp. 594–607, 2013.

[9] V. R. Chandrasekhar, D. M. Chen, S. S. Tsai, N.-M. Cheung, H. Chen, G. Takacs, Y. Reznik, R. Vedantham, R. Grzeszczuk, J. Bach, and B. Girod, "The stanford mobile visual search data set," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 117–122. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943568

[10] L.-Y. Duan, F. Gao, J. Chen, J. Lin, and T. Huang, "Compact descriptors for mobile visual search and mpeg cdvs standardization," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, May 2013, pp. 885–888.

[11] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 304–317.

[12] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[13] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.

[14] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," in *ACM Transactions on Graphics (ToG)*, vol. 23, no. 3. ACM, 2004, pp. 303–308.

[15] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2274–2282, 2012.

[16] S. A. Chatzichristofis and Y. S. Boutalis, "Cedd: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval," in *Proceedings of the 6th International Conference on Computer Vision Systems*, ser. ICVS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 312–322. [Online]. Available: http://dl.acm.org/citation.cfm?id=1788524.1788559

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.