Crowdsourced Rotoscoping

Hanzhong Ye

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-159
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-159.html

June 3, 2012

Acknowledgement

We would like to thank Bjoern Hartmann, Joel Brandt and Jitendra Malik for their continuous supervision throughout the technical development of this work. We would like to thank Ikhlaq Sidhu and Lee Fleming for their supervision on the strategy development for this project. Finally, we would like to acknowledge the creative common image owners, whose images were used by us for this study.

Crowdsourced Rotoscoping

Hanzhong (Ayden) Ye

Abstract

Rotoscoping is a technology used to create a mask for an element on an image or a video so it may be composited over another background, which has been a slow, process-intensive and costly manual task, especially on mobile devices where the screen is smaller than traditional screens and thus the user interface is limited. Given the current state-of-the-art, people still cannot fully automate the process of rotoscoping by computer vision techniques. We solve this problem by leveraging the power of crowdsourcing, which refers to the act of sourcing tasks to a large group of people or a community online. Specifically, we design a system which consists of a client-side mobile application to collect photos and a server-side pipeline for crowdsourcing. The client-side mobile application provides a user interface for smart phone users to take, upload, manipulate and save their photos. The server-side automated pipeline is built on Amazon's Mechanical Turk platform to generate masks from online crowd workers for pictures uploaded by application users. By connecting mobile devices to online crowd workers, our system manages to produce masks for different inputs and allows application users to generate various special effects using these masks. Experiments show that our system can produce acceptable results within reasonable waiting time. We finally make a discussion based on the experimental data and the user feedback collected from our user study for the mobile application.

Keywords

Crowdsourcing, Rotoscoping, Image Matting, Mobile Application

Introduction

Rotoscoping refers to the process of separating foreground objects from background canvas in a given image or video clip. This technique has been widely used in the movie and photography industry and can be commonly seen in TV commercials, music videos and visual advertisements (as shown in figure 1). Traditionally, rotoscoping is performed by specialists. With the development of computer vision technology, it is possible to use professional software tools to accelerate this process when dealing with simple images, but such tools usually require learning and training process. In addition, rotoscoping work usually has to be done on a desktop computer because of the highly interactive manipulation process involved. When dealing with video, rotoscoping becomes an even more difficult and challenging problem because of the time coherence problem and the huge amount of process-intensive works involved than single image. Besides tedious labor work for rotoscoping video clips, sophisticated technique, prohibitive cost and usually very long turnaround time are also big problems. For example, it is common that the expense of rotoscoping a video clip of 30 seconds exceeds 15,000 dollars, and the process would take more than 200 hours to be totally finished.

Figure 1: Single Image rotoscoping and video rotoscoping [1][2]

Under the current state-of-the-art of computer vision technology, human beings are still much better at detecting foreground objects from background than computers. As a result, the most effective way to do

rotoscoping is still through human labor. With the rise of crowdsourcing, many traditional manual labor tasks can be spread fast to a vast number of online workers quickly. The process of crowdsourcing is the act of sourcing tasks traditionally performed by specific individuals to a group of people (crowd) through an open call. Currently there are many different online platforms for crowdsourcing, and Amazon's Mechanical Turk is one of these platforms. On Amazon's Mechanical Turk, requesters can post task requests and workers can work on these assignments, which are called "HITs", to get paid. Developers can easily use the platform's well-developed APIs and payment system to integrate the platform with their own projects for their different needs.

Our approach to solve the rotoscoping problem is to take advantage of Amazon's Mechanical Turk platform to spread rotoscoping tasks to a group of online workers, who can then use the web-based tools we developed to process images by painting out masks on a given canvas. By doing this, we can leverage the power of crowdsourcing to reduce the cost and turnaround time of rotoscoping process for images. We further develop our idea by integrating our system with mobile applications: we use our automated pipeline on Amazon's Mechanical Turk as a back-end system of a mobile application (CrowdBrush) which people use to upload pictures onto servers to be processed. After the picture has been rotoscoped by the online workers through the pipeline, mobile application users can download the masks to add special effects to the pictures they upload (an overview of the system design is shown in figure 2). We did both cost and accuracy test on the back-end pipeline and user experience study for the front-end Android application, after which we refined our design to improve the system based on these data and feedback.

Figure 2: System design overview of crowdsourced rotoscoping system, including a client-side mobile application on Android (upper half) and a server-side automated pipeline for crowdsourcing (bottom half)

In this paper, we introduce the design of our system, including a back-end pipeline and a front-end mobile application. We introduce the specific technology we select such as PHP, Python, MySQL, Flash, Java and some details on our implementation. We analyze the experimental data and user feedback obtained from our experiments and user study, which help us better understanding how to design a good product with intuitive user interface. We discuss the advice we collected from two experts in the mobile and crowdsourcing industry, upon which we further discuss the possibilities to improve the application and approaches to push it to vast mobile application users. We finally draw to the conclusion that our system provides a feasible approach to produce acceptable rotoscoping results within reasonable waiting time. We also discuss the valuable lessons we have learnt from this project.

Related Work

Theoretical Work on Crowdsourcing

The term "crowdsourcing" is a portmanteau of "crowd" and "outsourcing," coined by Jeff Howe in a June 2006 Wired magazine article "The Rise of Crowdsourcing" [3]. After this concept was established, many works have been done to solidify the theoretical foundation of crowdsourcing. Quinn et al [4] concluded on the classifications and terms that emerge in the area of crowdsourcing and human computation, and this work serves as a constructive guidance for later research in this area. Von Ahn et al [5] introduced the concept of GWAP (Game-With-A-Purpose), giving up an insight of the power of games to bring motivation for game players to participate in human computation activities. Little et al [6] analyzed iterative and parallel human computation processes systematically, and they concluded different models for iterative and parallel process, and divided group-sourced tasks into creation tasks and decision tasks. All these theoretical research helps us better understanding the process of crowdsourcing.

Real World Application of Crowdsourcing

Although currently there is not an identical research to our work, there have been many studies and products on both crowdsourcing and image-recognition sides. A famous example is ReCAPTCHA, a crowdsourced OCR (Optical Character Recognition) project done by researchers at Carnegie Mellon University [7]. CAPTCHA is a common program on Internet which asks a user to recognize text in a complex image to validate the user is a real human instead of a computer. ReCAPTCHA combines OCR with CAPTCHA, so a user recognizes one word from a real document and one computer-generated word simultaneously, as shown in figure 3. Another crowdsourced image-recognition project was done when famous computer scientist Jim Gray was reported missing during a solo sailing trip in 2007. To search for location clues, thousands of satellite images were posted on Amazon’s Mechanical Turk website for

crowd to search for his boat in those images [8]. These real world applications show us the power of crowdsourcing, as well as the methodology used to design a crowdsourcing system.

Figure 3: ReCAPTCHA crowdsourced OCR system [9]

Mobile Image Manipulation and Recognition Applications

On the mobile side, there are many applications for image sharing, processing and recognition on mobile application platforms such as Android and iOS, which we hereby classify into two categories. The first category helps people to edit and add special to photos and share them with friends. Photo sharing application Instagram provides a free and simple way to add special filters on photos and share them in its own sharing network with more than 20 million users [10]. TouchRetouch is an image processing application enabling people to remove unwanted content or objects from a given photo. People can mark the unwanted items and it uses an algorithm to delete those parts and recover the original background [11]. Photosynth is another example which provides a set of tools for capturing and viewing the world in 3D with a panaroma constructed through a special technique of 3D reconstruction in computer vision [12]. VizWiz is an iPhone application that allows blind users to receive quick answers to questions about their surroundings. VizWiz combines automatic image processing, anonymous web workers, and members of the user's social network in order to collect fast and accurate answers to their questions [13]. CardMunch is an application powered by LinkedIn which leverages a real human workforce that manually transcribes each and every card a user submits into digital identifications on LinkedIn professional network

[14]. These mobile applications show us the potential of combining mobile devices with crowdsourcing system. We also learn about the design of user interface from these mobile applications (some of these examples are shown in figure 4).

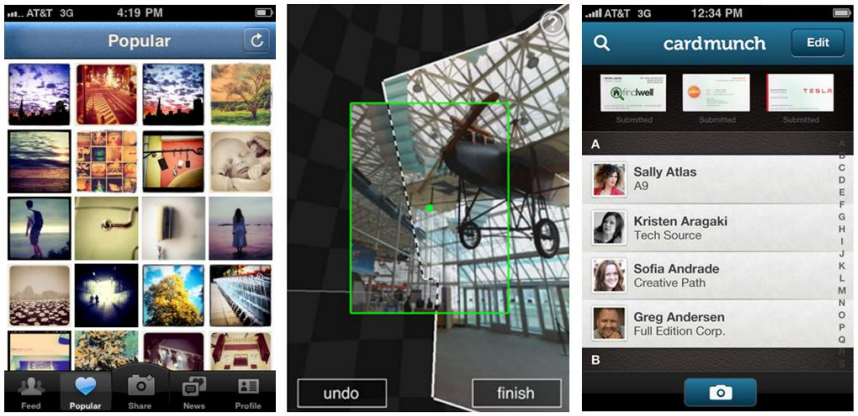

Figure 4: Image sharing, manipulating and recognizing applications (Instagram, Photosynth and CardMunch)

Methodology

Our goal is to design and develop a system that can produce satisfactory image masks and related special effects for pictures uploaded by mobile application users, through an automated crowdsourcing pipeline. In order to do this, we try to seek for the best system design which connects online workers with rotoscoping task requesters, as well as to streamline the work process and to evaluate the quality of works. We accomplish this by design different types of systems and run efficiency and quality test for these different prototypes. After making a decision on the design of the back-end crowdsourcing pipeline, we design and implement the front-end mobile application and then launch user study it with real world application users. We collect feedback from these users to improve our system. We also collect opinions from experts in the mobile and crowdsourcing industry. In the following, we will specifically describe each part of our methodology.

In the first semester, we focused on the design, implementation and system test of the back-end automated crowdsourcing pipeline. After learning from several previous studies on the methodology of system design for a crowdsourcing workflow, we proposed an overall design of our pipeline, as showed in figure 5.

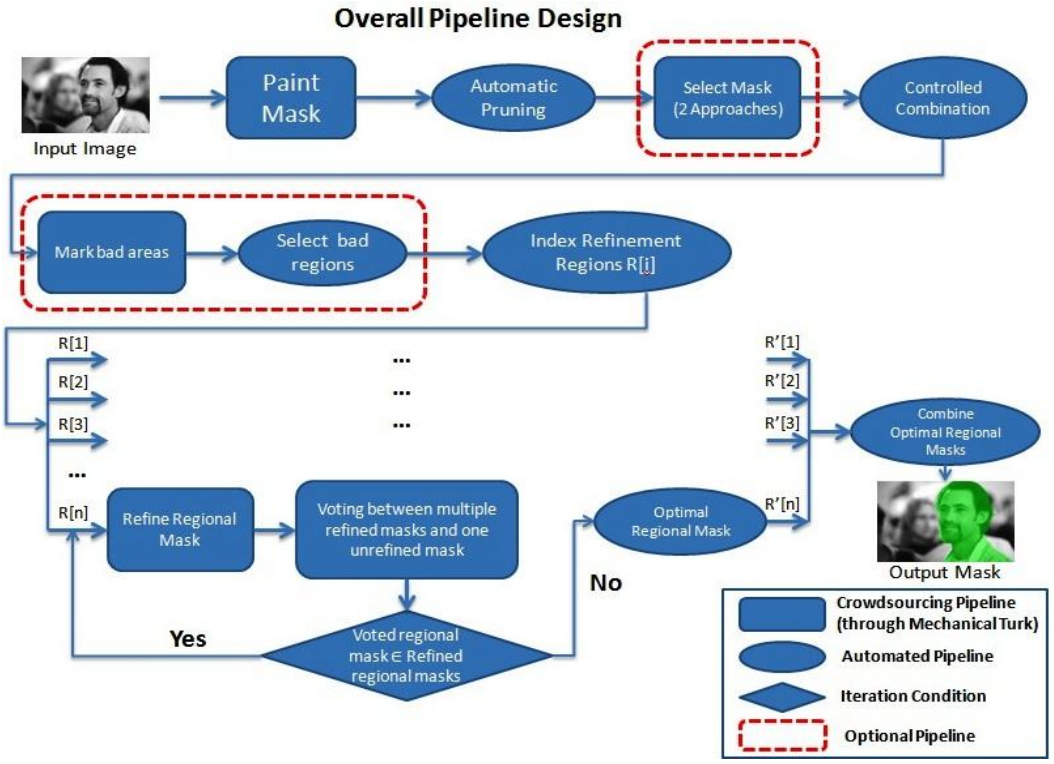

Figure 5: Original system design of the automated crowdsourcing pipeline. The entire pipeline can be subdivided into four sub-systems, which are 1) Obtain initial masks and prune, 2) Selective combination of initial masks, 3) Refinement region identification and 4) Iterative regional refinement.

On a high level, we decided to crowdsource two types of tasks – drawing of masks and voting to select best masks from a bunch of submitted masks. All other tasks such as pruning of outlier submissions to decide who gets paid for the task, breaking an image into smaller pieces for refinement and combining work of multiple crowd workers are automated. As for the platform, we chose Amazon's Mechanical Turk platform because it provides a well-structured crowdsourcing environment where individuals can easily recruit workers to work on small tasks, and developers can easily integrate their own projects with the platform. Built upon Amazon's Mechanical Turk platform, our system takes uploaded pictures as inputs,

acquires initial masks from multiple workers, selectively combines their output and iteratively improves the result by asking workers to identify and refine problematic regions.

For the implementation, the web interface allowing workers to create masks and mark regions on an image is developed on Flash, and we show workers examples of bad works before they start working, as shown in figure 6. The voting tasks and back-end database are handled by PHP and MySQL. The tasks are posted on Amazon's Mechanical Turk by using Python based "Boto" toolkit. The scripts for mask pruning, combination and region generation are all written in Python, using the Python Image Library.

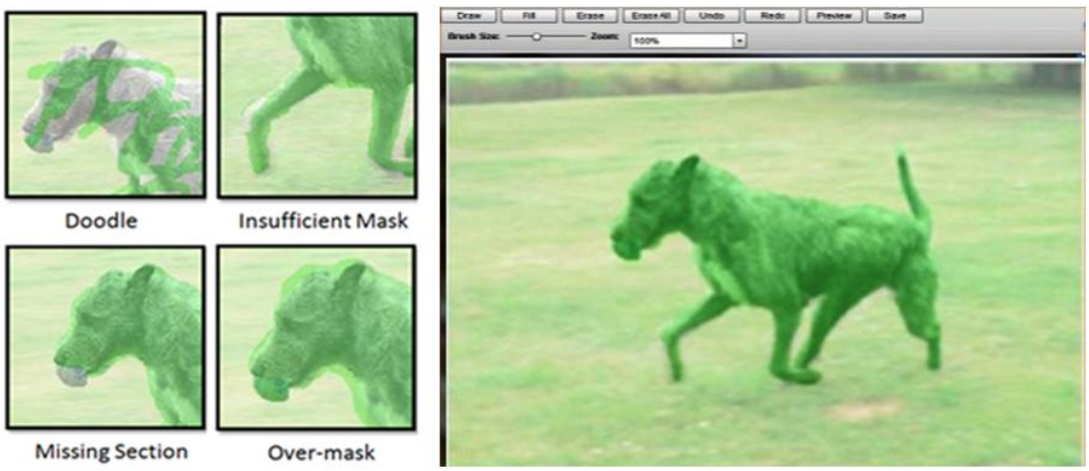

Figure 6: User interfaces for online crowd workers and common errors

In this model, the biggest assumption we have to test is whether or not we need an iterative regional refinement process in our pipeline. To test our assumption we ran a series of experiments to test the efficiency and results' quality of the pipeline after implementation. However, the results showed that the iterative refinement section is quite redundant and doesn't contribute much to the overall performance of the system. This led us to a much simpler design for our final model as showed in figure 7. We will discuss these experiments in the discussion section.

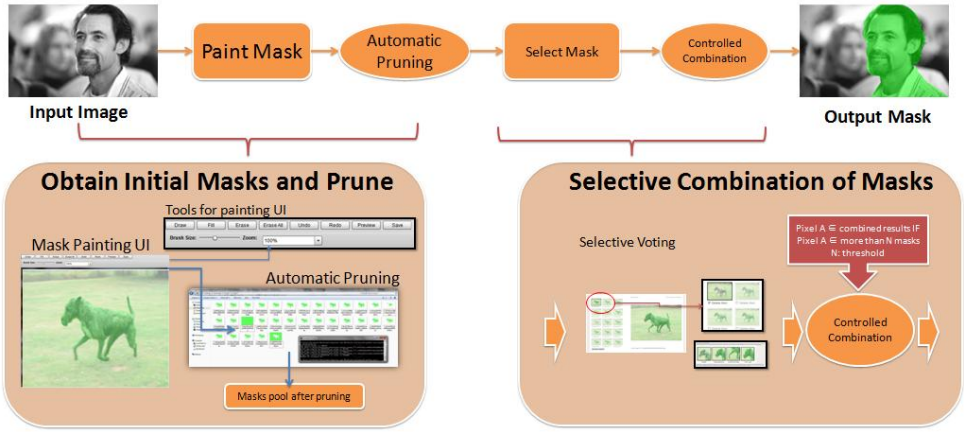

Figure 7: Final system design of automated crowdsourcing pipeline

In the second semester, we continued our work on developing a mobile application on Android platform, which enables smart phone users to take photos and upload them to servers. Learning from some existing mobile applications which we have discussed in the related work section, we decided to develop an application which can leverage the power of crowdsourcing, and provide end-users a series of special effects based on the foreground masks generated, and finally enable them to share these edited pictures with their friends through their social network. We also learn from these applications about the design of user interface and user experience, which gives us reference and guidance on the design of CrowdBrush.

Before implementation, we produced several mock-up sketches to walk through the user experience and design the user interface, some of which are shown in figure 8. We learn from these sketches as well as from the design process that it is important keep the design of user interface as intuitive as possible to bring better user experience and shorter learning time for application users.

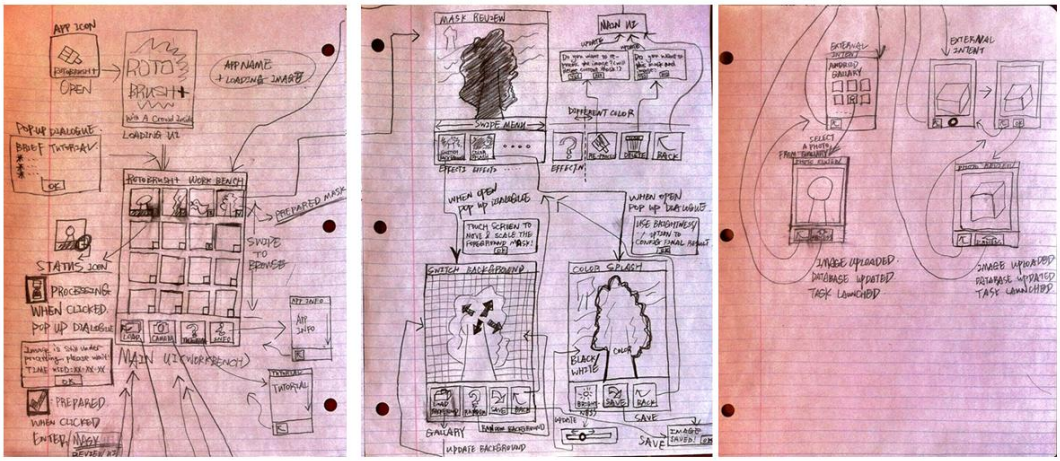

Figure 8: Mock-up sketches for the application user interface

Based on our design, we implemented our front-end application using Java and Android SDK. We firstly turned mock-ups into XML codes that construct the layout element, and then implemented the features we designed for the application. We name this application “CrowdBrush”, which provides the following user experience and features: when a user opens CrowdBrush application, she can register a user ID or log in with her ID if she has one. After log-in, she goes into an interface which we called the “WorkBench”, which showcases all the pictures both being processed (with an hour glass icon) and prepared for special effects (with a tick icon). She can either take a picture using smart phone camera, or load a picture from her album to be placed into the WorkBench. She will be asked to verify her uploading before she presses the “process” button, after which a task will be generated on the server side, and a crowdsourcing task will be triggered on the automated back-end pipeline. The status of the task will remain “processing” until the rotoscoping process is finished and a result is downloaded into the client-side smart phone. When a final mask is downloaded, the application will send a push notification to the smart phone user, which notices her to check her mask in CrowdBrush. When the user goes back to the application, she can review her mask, and decide if she is satisfied with that result (so she can press on “add effect” button to add different types of special effects with the generated mask), or choose to rework (after which the

application will resend the initial image onto the server and re-trigger the crowdsourcing pipeline) or delete the task.

CrowdBrush provides three types of special effects to the application users (as shown in figure 9). The first one is called Switch Background, which users can use to switch the background of the original picture. Users can load pictures from their native smart phone album, while we also provide a pool on the server where users can randomly download background pictures. They can also adjust the position of the foreground object by touching and dragging the object directly on the screen. The second feature is called Color Splash, which casts a special effect that only foreground object or background canvas will be rendered with color, while the other will be rendered in black and white. User can adjust the brightness of either the foreground or background. The third feature is called Sticker, which generates a sticker-like effect, with an additional sticker boundary produced along the edge of the foreground. After a user finishes editing, she can save the edited picture into her smart phone album, or share it through her social networks such as Facebook. Some selected screenshot of the user interface we have designed and implemented are shown in figure 10.

Figure 9: Overview of CrowdBrush image editing features. (a) Original Image (b) Switch background (c) Color Splash (d) Sticker

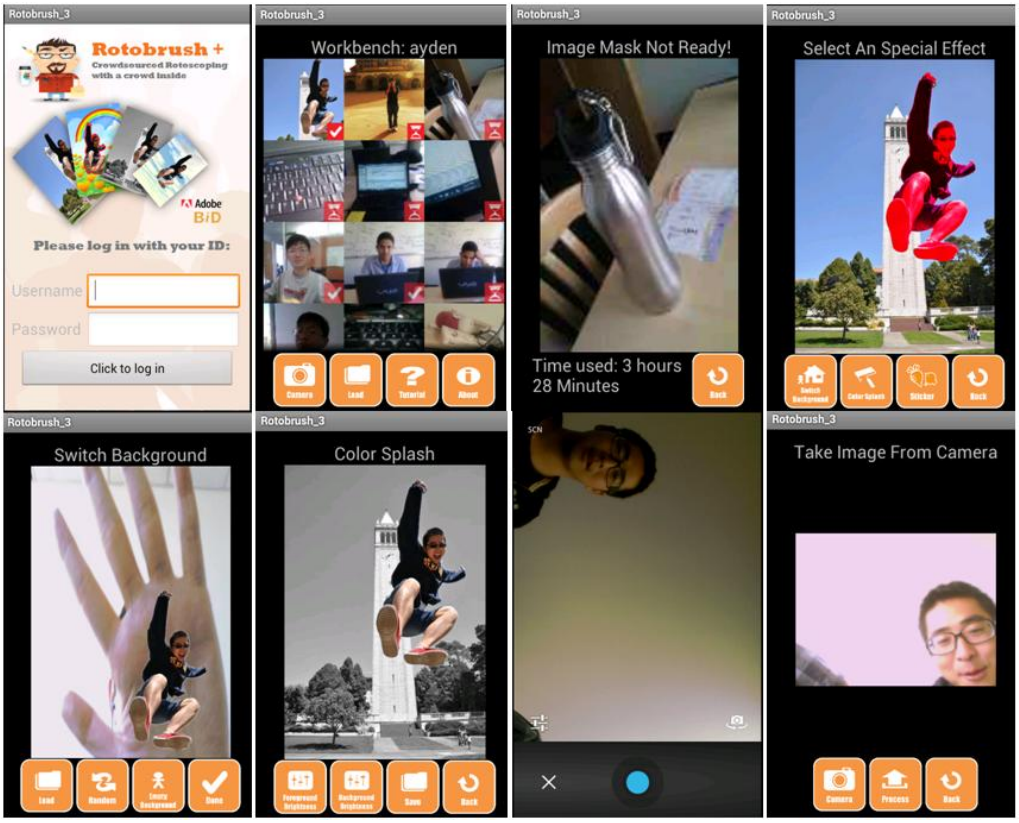

Figure 10: Screenshots of selected user interfaces of CrowdBrush application. From top-left to bottom-right: (1) Login interface (2) Workbench interface (3) Waiting tasks (4) Select special effect (5) Switch Background (6) Color Splash (7) Take photos (8) Upload and process photos

We implemented CrowdBrush using Eclipse IDE, and Java programming language. We utilized a series of SDKs and integrated tools such as Android SDK, Facebook SDK, Mosquito (an integrated toolkit to generate push notification), etc. We also tested the application and improve our code to enhance software usability. After implementing and testing both back-end pipeline and front-end application, we integrated two parts together. For integration purpose, we added more triggers in the back-end system so that it became fully automated by itself, and able to produce masks whenever receiving the request from front-end application.

To evaluate the user experience and usability of CrowdBrush, we launched a user study to test our hypothesis and get feedback. Before the user test begins, we collect some information about user's habit

of using smart phone and applications. Then, all users are asked to walk through the following steps after opening the application: Login -> Take a picture -> Save the picture -> Submit the picture and start processing -> Wait till the mask comes back (which takes 2~6 hours) -> Change background using the Switch Background feature-> Manipulate with the foreground objects until reaching their satisfaction -> Save the image -> Try Color Splash feature -> Use Sticker feature -> Test ends. After the walkthrough, we asked them to talk about their user experience and provide some feedback and suggestions.

Samples of survey questionnaire we used for this study are shown in figure 11.

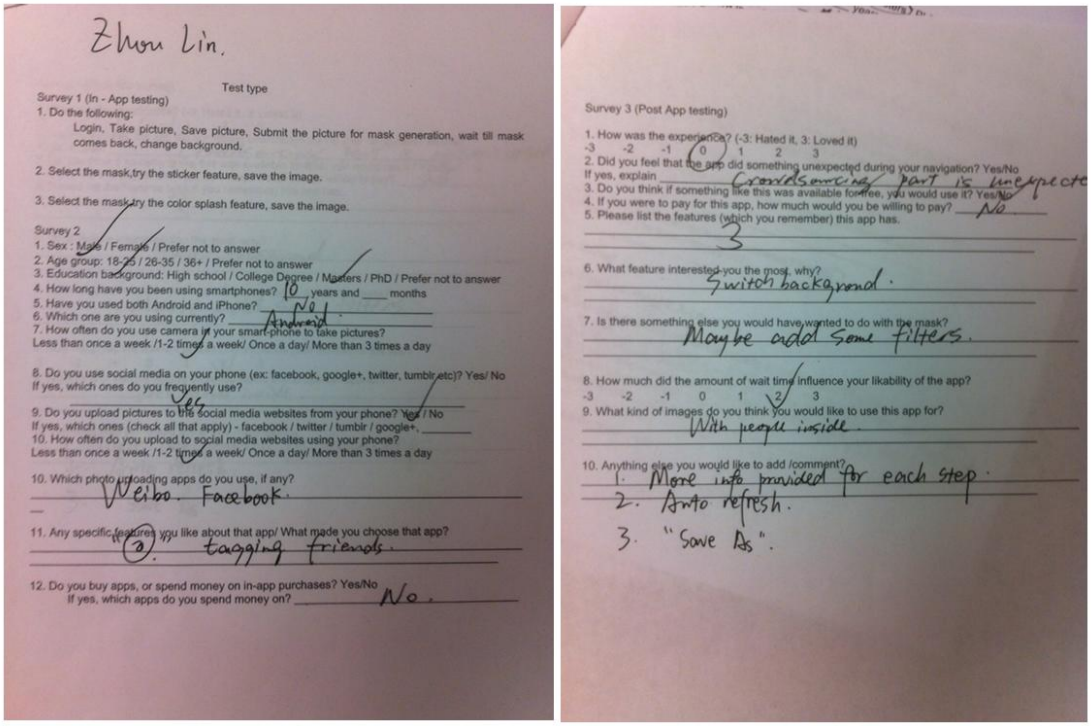

Figure 11: Sample of survey questionnaires used for user study

The details and results of this user study will be discussed in the discussion section. In addition, we also did two expert Interviews, which will also be covered in the discussion section.

Discussion

After we have finished implementing the back-end pipeline, we ran a system test to evaluate the efficiency and results' quality of the automated crowdsourcing system. In our experiments, we attempted

to validate the need of a two step pipeline (initial masks collection and iterative regional refinement). We took 2 images of varied complexity levels: Easy, Medium and Hard (as shown in figure 12). We took each image 3 times through the pipeline and evaluate the results obtained.

Figure 12: Pictures tested in crowdsourcing pipeline

Results without iterative regional refinement are shown in figure 13. We noticed that there is no significant level of improvement in the quality of masks when we included the regional iterative refinement process (as shown in figure 14), which cost us 5 times more on spending and nearly 48 hours of time as compared to 6 hours when using a simpler pipeline (as shown in figure 15). Based on this, we decided to cut off the iterative refinement process to have better control over time and money cost for each task.

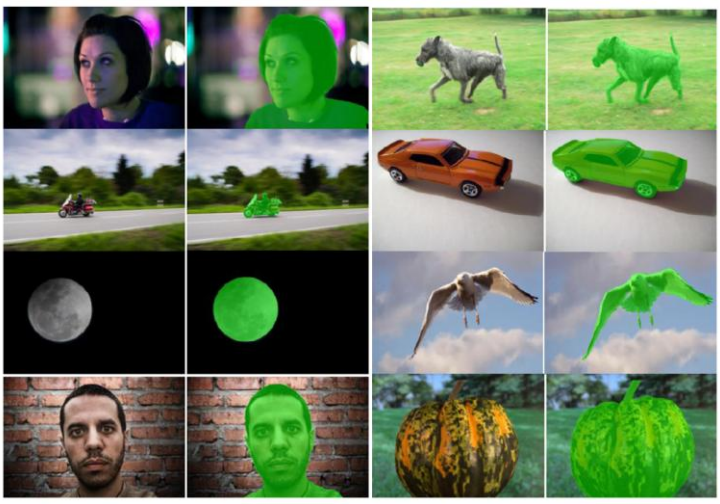

Figure 13: Mask generated without iterative refinement process

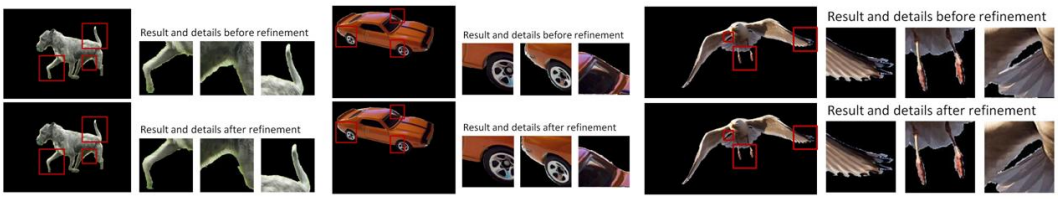

Figure 14: Quality difference between the results for pipeline with/without iterative refinement

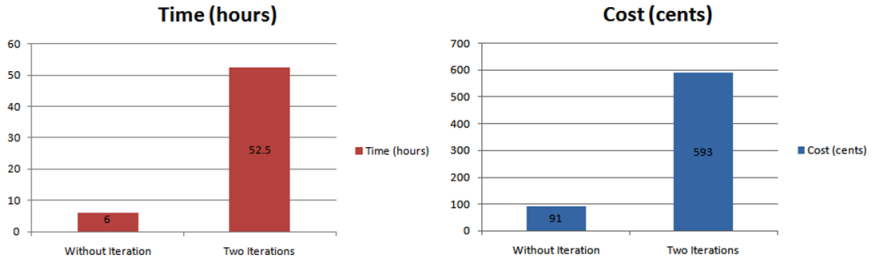

Figure 15: Time and money cost difference for pipeline with/without iterative refinement

As another validation of our crowdsourcing approach, we compare it with one computer vision image matting approach—GrabCut. The Grabcut algorithm models foreground and background color distribution by Gaussian mixture model and models image matting problem as binary classification of each pixel as foreground or background. Figure 16 shows the results we produced from two different approaches. In (a), we find both crowdsourcing pipeline and GrabCut can generate satisfactory foreground masks. In (b) however, Grabcut is obviously confused because there is no obvious foreground object in the scene, while our crowdsourcing pipeline can still generate mask making sense semantically. Again in (c), GrabCut fails to capture the white font on the jacket because of its color is similar to the white wall in background, although it does create a better boundary of the person than our crowdsourcing pipeline. We also iteratively refine the output of GrabCut by letting user to specify where goes wrong, but we find GrabCut still fails to capture the font after this iterative refinement. In (d), GrabCut fails to distinguish the jeans and the carpet because the similarity in color. The result after iterative refinement eliminates part of the jeans from the foreground, which shows that iterative refinement sometimes even leads to worse results. In short, we conclude that although computer vision

approach can perform well in certain scenes, it can still fail in more complex real world scenes. On the other hand, our crowdsourcing pipeline is able to generate reasonable results more consistently.

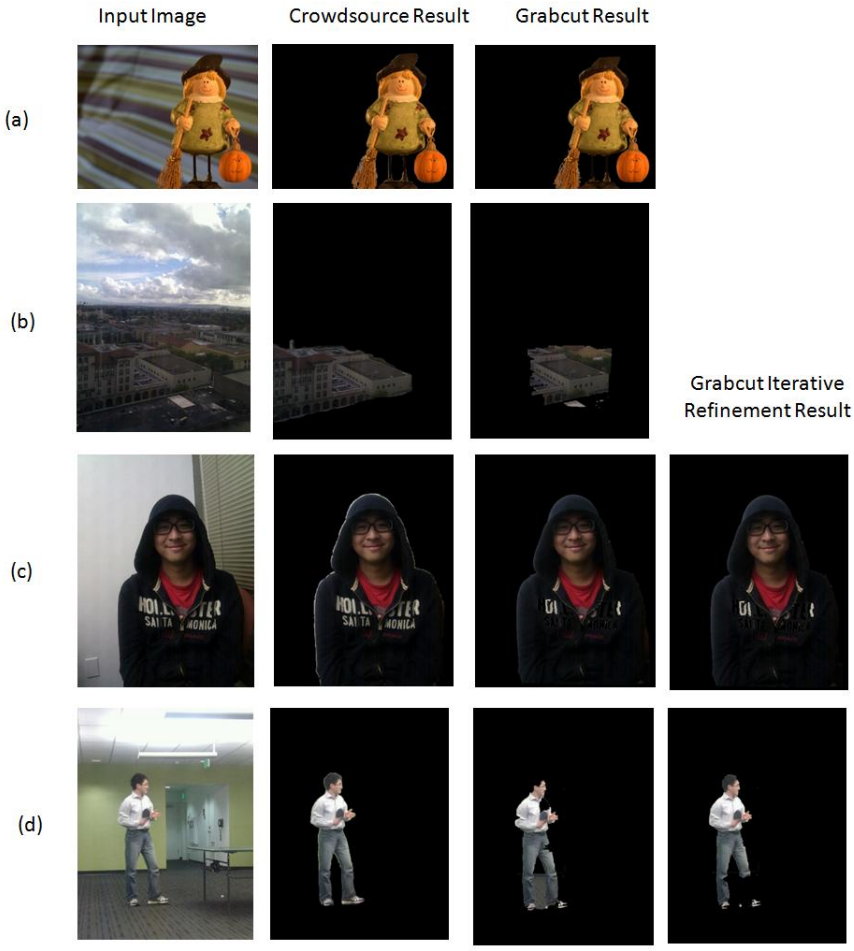

Figure 16: Comparison between crowdsource results with Grabcut result. The first column is input image; the second column is the result mask of our crowdsource pipeline; the third column is the result mask of GrabCut; the fourth column is the result of iterative refined GrabCut

In the second semester, we focused on the mobile application and its integration with our automated pipeline. During our implementation, we did two expert interviews to get professional comments and advice from experts in the mobile and crowdsourcing industry. The first interviewee is Raymond Wei, Co-founder & CEO of Weaver Mobile, Inc, who has years of working experience in the mobile industry, especially in photo sharing applications. The second interviewee is Philipp Gutheim, Co-founder & COO of MobileWorks, a company that provides a platform similar to Amazon's Mechanical Turk with more intelligent features helping improving crowdsourcing process.

Raymond has given us a bunch of useful suggestions, some of which we have later developed as new features in CrowdBrush. First of all, we took his advice on the integration with Facebook API. Although we didn't use SSO to generate user ID, we implemented sharing features into our application. Secondly, we took his advice to pull data storage onto the server side, while keeping a local cache on the client side. Thirdly, we learnt about the possibility to use HTML5 to quickly produce applications on multiple platforms, although we did not actually write another project in HTML5. We learnt about several important issues such as integration with social network, Client/Server architecture design, software usability issues, and so on. We also learnt a lot from our second interview with Philipp. Firstly, we learnt about some alternative platforms besides Amazon's Mechanical Turk, such as MobileWorks (as showed in figure 17), and their strength on quality and turnaround time control, as well as its potential to further cut down the cost for each task. Although we did not implement our system into MobileWorks platform, we understood the potential benefits it would bring to our project. Secondly, we have learnt about the idea of doing crowdsourcing tasks on mobile device, and the potential social value of this concept. Thirdly, we obtained some good suggestions on how to make people more willing to pay. Learning from companies like PicPlum, we understand that we can do more value-added service for users, such us in-depth editing according to their description, photo printing and delivering to make them more willing to pay. Last but not least, we have learnt that the real value of crowdsourcing is to provide a more efficient task management channel to better leverage human resource. We have learnt that eventually the balancing problem between crowd labors versus artificial intelligence is going to become an optimization problem, which looks for an optimal combination of human and computer labors, with better defined value and clearer goals set for crowdsourcing.

Figure 17: MobileWorks crowdsourcing platform, an alternative to Amazon's Mechanical Turk

After implementing CrowdBrush and collecting advice from the experts, we launched a user study for CrowdBrush application, as mentioned in the methodology section. The main goal of our user study on the application is to test the usability and collect user feedback for user experience and design. Following is our study results and discussion.

In total, we have 20 participants in our survey, 90% of which are Master students at UC Berkeley. 17 are males and 3 are females, with age ranging from 18 to35. With an average age of 2 years using smart phones, 60% of them reported using iPhone's iOS and 40% reported using Android as application platform. Around 80% said they use Facebook on smart phones for sharing photos. Besides Facebook, other tools are reported such as Instagram, Twitter, Weibo, Renren (Chinese Facebook and Twitter). The average frequency of uploading photos is 1-2 times/week. As for purchasing behavior, around half of them reported having purchased applications before (mainly games), while only 10% said they have made in-application purchases. For the cost of each task, on average a mask cost $0.40 and takes 3.5 hours on Amazon's Mechanical Turk (calculated for the 60 tasks). When we asked about users'

willingness to pay for CrowdBrush, 20% said they would be willing to pay up to 1.99$ for a one-time purchase for this application, while none of them showed willingness to pay for each processing.

To evaluate user's satisfaction with their user experience, we asked them to evaluate their user experience with a Likert scale ranging from -3 to +3, and on average they rated 1.5 for their user experience. We also asked users to evaluate the influence of waiting time to their willingness to use the application, and most people selected 3 (maxim) to describe the influence of turnaround time to their willingness to use the application, which shows that people are highly sensitive to the waiting time. For the features we provide, 75% of the users found Switch Backgrounds features to be interesting. 10 users used this feature twice to improve their editing. Some reported that they liked the idea of being able to interact with the masks, and wanted more interactive features like the ability to scale up/scale down the masks. Some suggested adding other interesting features, for example, letting them work on the masks of their friends. Some suggested we can add some pre-saved masks in the application for people to play with, which we consider to be a useful advice because some users complaint that they wanted to know what they can do with the masks while they were waiting for their photos to be processed.

From this survey, we learn that most users considered the application to be not hard to navigate with, and understood well the idea of the application and learnt fast in playing with it after a brief introduction. In terms of the performance, the first concern for most people is the turnaround time. Currently it takes 2~6 hours before one can download the masks, which is still far beyond people's expectation. Most people said that they would at most expect to wait for 1~2 minutes. Another issue is that the users didn't show

much willingness to pay for this service by each uploading, which would be a big challenge for us if we want to make this application a real product and push it to the market.

Conclusion

During this project, we have explored the possibility of leveraging crowdsourcing approach to handle rotoscoping problem. Finally we have several important conclusions for this project.

Firstly, with a well developed crowdsourcing platform and a group of well trained crowdsourcing workers, we can design an effective workflow and an automated pipeline which produce acceptable results within a reasonable turnaround time. The crowdsourcing pipeline we implemented is able to tackle inputs with various complexity, and provides more consistent outputs than computer vision approach. Secondly, integrating mobile clients (such as mobile applications) with automated crowdsourcing system is feasible, and it provides many possibilities for people to handle tasks which are traditionally not easy to handle with on a mobile device. Last but not least, the design of the back-end automated pipeline for crowdsourcing significantly influences the efficiency and turnaround time of the workflow, while the design of the front-end application's user experience and interface greatly influences user's real experience and their willingness to use the application. In short, design matters much.

However, currently our system still has several obvious shortcomings, which leads to further problems to be solved. For example, how to further cut down the cost of crowdsourcing process to a level that application user will be willing to pay for the service on a regular basis? How to further guarantee the

quality of the results? How to include more artificial intelligence into our current system to further accelerate the process? These are all problems to be solved in our future work.

One of other topics deserve exploration is more potential applications of a similar system but for other purposes. Since rotoscoping is only a particular application, we believe a similar system can be used in a much wider scope -- for example, editing, recognition, processing, evaluation for text and video and other forms of multimedia. We believe the power of crowdsourcing can be further leveraged in more fields than the area we have focused on.

References:

1. http://www.digitalfilmtools.com/ezmask/
2. http://effectscorner.blogspot.com/2006/01/rotoscoping-part-1.html
3. David Whitford (March 22, 2007). "Hired Guns on the Cheap". Fortune Small Business. Retrieved August 7, 2008.
4. Alexander J. Quinn, Benjamin B. Bederson. Human Computation: A Survey and Taxonomy of a Growing Field. CHI 2011.
5. Von Ahn, L. and Dabbish, L. Designing games with a purpose. Communications of the ACM 51, 8 (Aug. 2008), p. 58-67.
6. Greg Little, Lydia B. Chilton, Max Goldman, Robert C. Miller. Exploring Iterative and Parallel Human Computation Processes. HCOMP 2010.
7. ReCAPTCHA Project, Wikipedia. http://en.wikipedia.org/wiki/ReCAPTCHA

8. Katie Hafner. Silicon Valley's High-Tech Hunt for Colleague. February 3, 2007 http://www.nytimes.com/2007/02/03/technology/03search.html?_r=1&ex=1328158800&en=e58764b50c8a4508&ei=5090&partner=rssuserland&emc=rss

9. http://www.captcha.net/

10. Instagram, http://instagr.am/

11. TouchRetouch, http://itunes.apple.com/us/app/touchretouch/id373311252?mt=8

12. PhotoSynth, http://photosynth.net/

13. Vizwiz, http://vizwiz.org/

14. CardMunch, http://www.cardmunch.com/

Acknowledgements

We would like to thank Bjoern Hartmann, Joel Brandt and Jitendra Malik for their continuous supervision throughout the technical development of this work. We would like to thank Ikhlaq Sidhu and Lee Fleming for their supervision on the strategy development for this project. Finally, we would like to acknowledge the creative common image owners, whose images were used by us for this study.