CrowdBrush: A mobile photo-editing application with a crowd inside

Suryaveer Lodha

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-136
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-136.html

May 30, 2012

Acknowledgement

I would like to thank Joel Brandt and Bjoern Hartmann for their continuous supervision throughout this work. Additionally, Joel Brandt provided us server access and funds to run experiments on Amazon Mechanical Turk and Bjoern Hartmann provided us Android handsets (Google Nexus One) to develop an android application and conduct user studies. I would like to acknowledge the creative common image owners, whose images were used for this study. Also, I would like to thank our fellow classmates at UC Berkeley who helped us in initial user testing of the CrowdBrush application and gave us invaluable insights and opinions about our UI design and application features.

CrowdBrush: A mobile photo-editing application with a crowd inside

By

Suryaveer Singh Lodha

4th May 2012

TABLE OF CONTENTS

Abstract

As per a recent report from NPD, consumers now take more than a quarter of all photos and videos on smartphone[1]. Mobile photography is gaining popularity, but editing photographs on mobile devices is cumbersome. Almost all of the current photo manipulation apps provide standard, static and one-filter-fits-all suite of photo editing features. While sophisticated techniques to segment objects in images exist, they are limited to desktop computing for multiple reasons: mobile device touch screens make selection hard, mobile devices have small screens and these techniques require significant computing power. A robust semantic segmentation of an image can lead to interesting photo-editing features such as switch-background on the mobile device with minimal end user input. Traditionally, rotoscoping[2] has been a slow, process intensive and costly manual task. Computer scientists have tried to solve this problem through automation by developing computer vision techniques, but have failed to achieve satisfactory results for complex videos and images. Identifying if a pixel belongs to foreground/background element is a trivial task for humans; hence we address this problem by introducing crowd sourcing techniques. We have developed an android application for photo editing on top of a low cost crowdsourcing system. We use the crowdsourcing platform to add interesting effects to images taken by users which they can later share with their friends. CrowdBrush application thus provides a feature of custom photo editing, which is missing from the current photo manipulation applications in the market. Additionally, the system produces "human-verified" results as it relies on human (crowd) involvement in the process and guarantees an end product which is aesthetically pleasing. We compare this crowdsourcing approach to the automatic grabcut

[1] https://www.npd.com/wps/portal/npd/us/news/pressreleases/pr_111222

[2] Rotoscoping is the process of separating foreground element from background in a video/image.

computer vision technique and conduct a user study with 20 subjects for our mobile application. We see that crowdsourcing solution works better than grabcut for complex images. We also learn that speed at which we can generate the output matters more to users than quality. We observe that users love to be able to interact with the application. The fact that they could change the photo easily and add new effects provided a sense of amusement and fun to the users.

Introduction

Rotoscoping involves the creation of image masks to isolate object in videos. High quality rotoscoping is a labor intensive and expert oriented task since it entails developing complicated boundary masks on every single frame of a video clip for further compositing work, and the masks need to be temporally coherent between frames, which makes the process tedious. As per discussion with a Principal Scientist working on Adobe After Effects, currently production houses spend around $70 per hour for an expert rotoscoping artist. Computer scientists have long been researching to develop state of the art computer vision techniques (example - Rotobrush in Adobe Creative Suite 5.5) [14 15] to aid and even replace humans in this task. However, Computer Vision techniques usually work well for uncluttered scenes, but fail in complex scenes which involve furry objects, occlusion, motion-blur, mist, fire and complex camera animation. Crowdsourcing video rotoscoping might be a better solution as it will save a significant amount of time by parallelizing the task across all frames, while also keeping the cost substantially low (as low as 35 cents per frame based on pilot studies). One crucial hurdle is quality maintenance over the crowdsourcing platform. We have developed a system to deliver fast and reliable rotoscoping results by introducing crowd workers to deal with problems such as identifying boundary pixels, which cannot be solved by computer vision algorithms. We compare crowd

sourcing solution with results from a popular Computer Vision technique – grabcut. The goal is to leverage the power of crowdsourcing to address the need of human intelligence in this work.

In this work, we focus on generating a foreground element mask for a single image. In the adjoining image[3], we show an example for mask generation of dog to switch its background. To get crowd workers to perform such complex tasks, the task must be broken into sequences of smaller steps. [1] This work investigates how to construct a workflow for creating high-quality image masks. It is difficult to describe open-ended high level tasks such as asking users to "rotoscope this image" especially when the crowd worker has not heard of rotoscoping. We break down this complex task into a series of smaller tasks (such as drawing a mask on an image), which can then be solved effectively on crowdsourcing systems.

Design Goals

While designing various user interfaces, the goal was to develop, test and iterate towards developing interfaces, which were intuitive and useful for the end user (on the android application) and the crowd-workers (for identifying foreground object in the image). We envisioned a regular scenario of the application use as the following (Figure 1) – an Android Application user uses the application to capture an image and submits it for processing via a push of the button. The submission triggers the backend pipeline which instantaneously creates a new HIT on Mechanical Turk, where crowd workers will be able to see the image in paint like interface and draw a mask on the foreground object in the image. The backend pipeline

[3] http://sharingcentre.net/forum/242782-amazing-mask-plugin-adobe-photoshop-cs3/

integrates work of multiple workers and develops a final mask for the foreground object in the image. A push notification is delivered to the user upon generation of the final mask, stating that uploaded image has now been processed. The user can now download the mask, use various features on the application to add interesting effects and save the image on the phone.

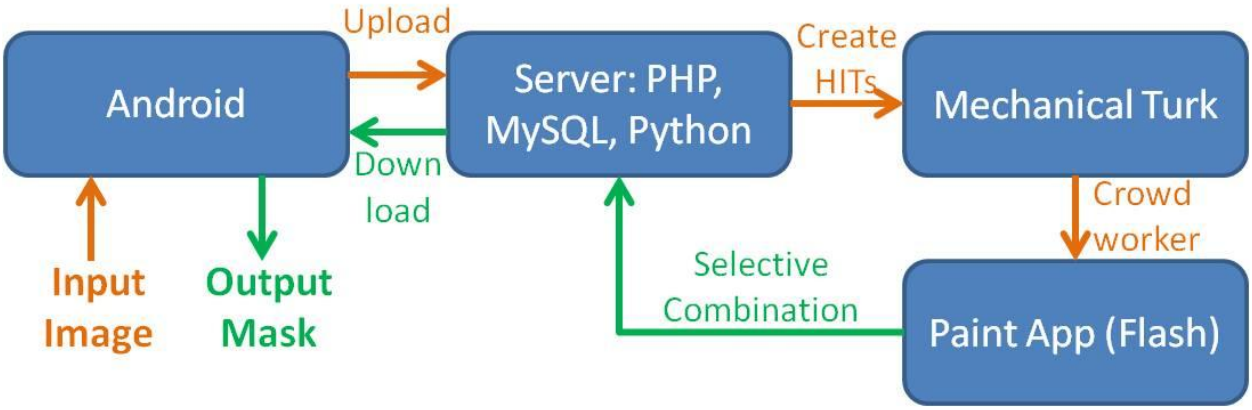


Figure 1: A diagram illustrating a typical usage of the CrowdBrush

As the user interface for rotoscoping (paint interface to generate foreground object masks) would be used by many crowd workers with varied backgrounds, it was vital for us to make it easy, yet efficient to have higher worker response for the tasks. Existing rotoscoping software is designed for (intended for use by) trained rotoscoping artist and involve complicated tools for curve editing. As crowd workers have no prior training to use such tools, it requires a substantial effort on worker's part to learn such tools. Therefore, our first focus was to develop a user interface which can be easily used (and learned) by untrained workers and allowed them to develop masks on a single image. We evaluate the ease of use and effectiveness of the paint interface by tracking the number of rejected submissions and calculating the average time spent by a crowd worker on the paint interface. We keep the interface simple and clean by only keeping tools which aid the majority of workers in quickly completing the task.

As online workers may misinterpret the task, try to shirk by submitting inferior solutions, or may not be suitably skilled for the task at hand, techniques for quality control are essential for achieving reliable results.[2] Hence, after collecting results from users, we focus on integrating useful submissions from crowd workers to achieve a high quality result. Here, we combine results from different workers into one single mask which covers most of the foreground object based on a voting among the crowd workers. We tested techniques to further increase the accuracy of the mask by visually defining subareas in masks and asking workers to iteratively fix them to converge towards a high quality solution, but we found such methods ended up increasing the time and cost exponentially for an image. We decided to use a non - iterative process with quick turn-around time, as it suits the requirements for a mobile application.

The rest of this paper is organized as follows: next section summarizes related work, followed by a detailed discussion of the pipeline design, evaluation plan and experimental results.

Related Work

In this work, my main contribution was developing a stable and efficient crowd sourcing solution for an abstract task such as identifying foreground object. I also contributed towards developing an easy to use paint like interface which crowd-workers use.

Crowdsourcing

Crowdsourcing has lead to the emergence of many real world applications. Quinn et al [3] classify existing human computation systems to provide insights towards future research work in human computation. A crowdsourcing system Soylent [4] which utilizes online resources to improve writing efficiency and quality serves as a good illustration of a real world application. Soylent discusses implementation of tools such as Shortn and Crowdproof, which successfully

group-source a highly abstract task of text editing and grammar correction. Soylent also contributes a Find-Fix-Verify model where crowd workers find and fix errors in text themselves. This model helped us in developing an iterative pipeline to iteratively refine results. Von Ahn et al introduce the concept of GWAP (Game-With-A-Purpose) [5], which gives insight into the motivational power of games over players who participate in human computation activities. Little et al. [6] systematically analyze the performance and trade-off between iterative and parallel processes of human computation. Spiro et al [7] designed a crowdsourcing application of gesture annotation, which performs video labeling. They point out that the design of the pipeline could consist of both human works and computer. They identify the design of user interface for crowd workers as the most important factor for success on the crowdsourcing platform.

Computer Vision approaches and related Image Selection Interfaces

Wang and Cohen [8] give an overview of current performance of computer vision algorithms on image and video matting. They conclude that while computer vision algorithms do a good job on a single image (such as Chung et al. [9] and Sun et al. [10]) current performance of video rotoscoping algorithms is still unsatisfactory. Wang and Cohen [11], Bai and Sapiro [12] present scribble-based interfaces which require users to simply specify a few scribbles of foreground and background as input and the algorithm generates the foreground mask for the input image. It would be very interesting to compare online workers' performance with or without this interface. Interactive Video Cutout [13] offers a novel approach to treat video as 3D volume and provides painting based UI to indicate foreground object across space and time. We take inspiration from such paint based UI's and work towards developing tools which are easy and simple to learn by workers on the crowdsourcing platform.

CrowdBrush Pipeline Design

This section first presents a brief overview of the CrowdBrush pipeline, followed by a detailed explanation of front-end and back-end systems. In the end, the implementation section summarizes different technologies used in developing the entire pipeline.

Pipeline Overview

The pipeline begins when a user uploads an image with his/her smartphone for processing on the server. Once the image is uploaded on the server, an entry is created in the database for the image corresponding to the user id. Then the system creates HITs for crowd workers on Mechanical Turk where multiple workers create a mask for foreground object using the Painting UI (Figure 2). All the masks collected through the first step then pass through an automated pruning step which discards masks which differ significantly from the average submission. Workers then vote among this group of "good quality" masks to identify the top quality user submissions. The top masks from the previous step are then automatically combined to get a merged global mask. The combination algorithm to create the final mask is discussed in the back-end development section.

Front-end Development

A majority of the UI design work for the Android Application (Figure 3) was completed by other team members. I partly contributed to development of the online Flash based drawing application which allows crowd workers to draw mask on the source image. The drawing application provides basic drawing functions such as different brush size, zoom, and eraser. We also developed several additional functions to assist workers in creating better masks quickly. These functions are undo/redo, region fill, and result preview (Figure 2). We kept these functions in the final painting UI based on highly positive feedback received by these functions.

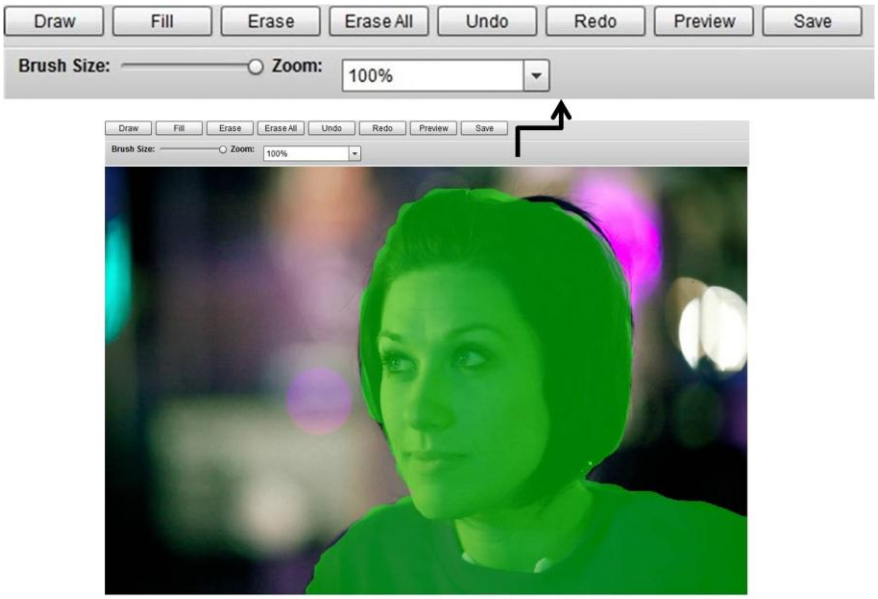
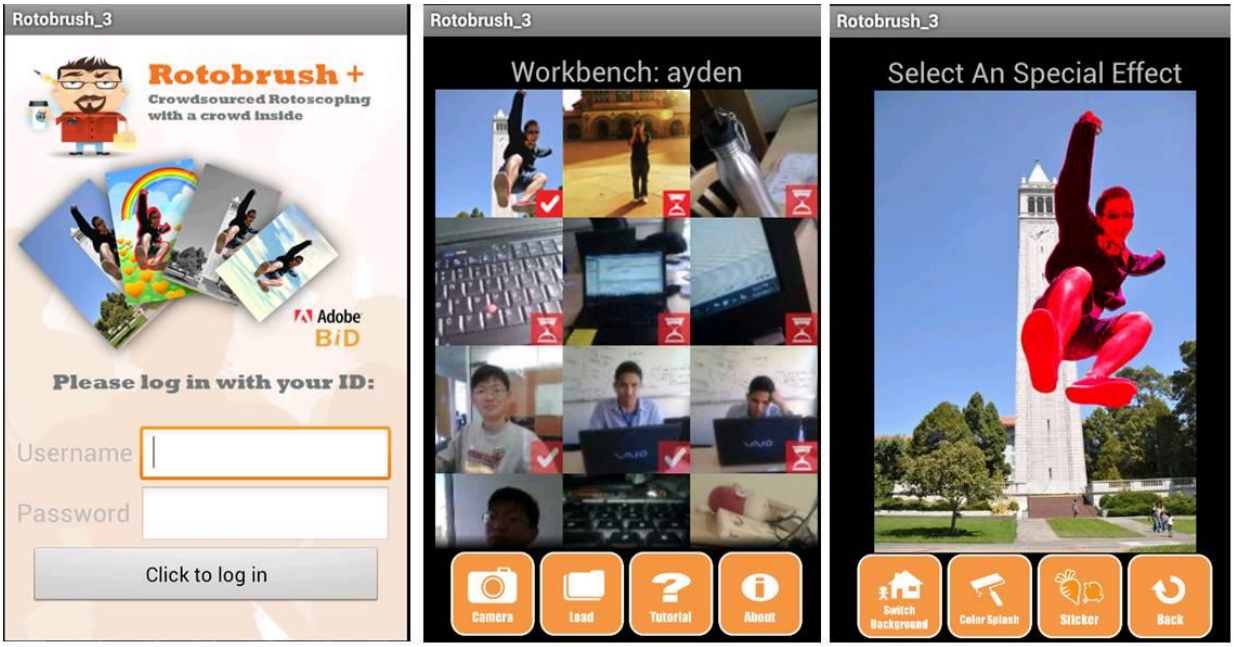


Figure 2: Painting UI with tools



Login Page used by user to log in the App

Workbench – shows the status of mask for images (Waiting/Done)

User can select different features of App here

Figure 3: Android Application UI Design – Login page, Work-bench, Effects Panel

Back-end development

There were three principal components for the back-end development: development of a system to interact with Amazon Mechanical Turk's interface, implementation of image processing algorithms on the server (for pruning and combination of crowd worker submitted masks) and a system which would interact between the server and the Mobile Application. I was fully responsible for building the system to interact with Amazon Mechanical Turk and the server and also handled implementation of the required Image Processing algorithms on the server.

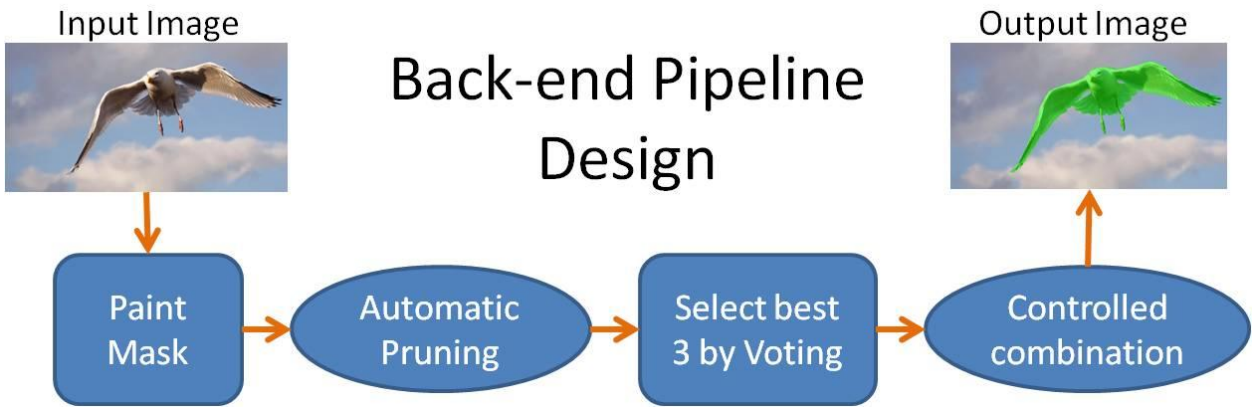


Figure 4: Back-end system for CrowdBrush. Squares - crowdsourcing, ellipses - automatic steps

The back-end system to interact with Amazon Mechanical Turk is developed using the open source boto[4] library. The entire platform is developed on Python 2.5. The system has the following significant features – ability to create HITs on Mechanical Turk whenever an image is uploaded on the server, ability to expire HITs early for any given image, ability to discard submitted masks based on heuristics automatically and develop the final result to be sent to the mobile phone from the server. I also implemented an automated payment system which would pay those workers whose work contributed to any stage of the pipeline. When a worker

[4] http://code.google.com/p/boto/

completes a task on Mechanical Turk, he is given a randomly generated 15 digit completion code, which is used by the payment system to pay a worker for the task which contributed in developing the final mask. Attracting enough workers on MTurk can be problematic as our tasks compete with many other requesters tasks. Prior research suggests that workers use the order of listing (e.g., number of available assignments) to decide which tasks to work on and prefer ones with more assignments. Hence, we optimize by posting 100 tasks, when only 7 are required. As soon as 7 are completed, the other remaining 93 HITs are automatically taken down. This search optimization technique helped in reducing average completion time from 6 hours to 3.5 hours.

After obtaining multiple results, we propose a heuristic method to audit how workers perform. We calculate the median of the number of foreground pixels marked in all submissions and those which significantly differ (threshold = 10%, based on pilot tests) from the median are discarded (Figure 5). We choose to use median over mean because median is more robust to outliers.

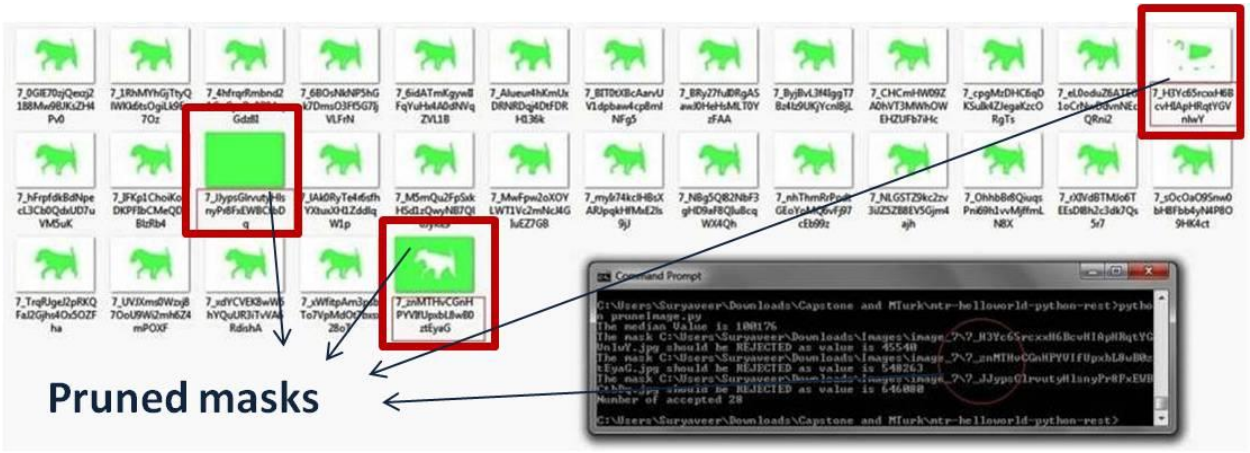


Figure 5: The pruning step – Inconsistent submissions (red box) are pruned automatically.

The Image processing algorithms were implemented using the Python Image library. We create a voting task on Mechanical Turk where crowd workers identify best masks to be combined for us. We provide a user interface for workers to select the best masks (Figure 6). We tried absolute and comparative voting schemes, but both gave us same results. In the comparative voting scheme,

workers vote to keep the best 3 masks by comparing them with all the good masks. In the absolute rating scheme, workers independently view one mask at a time and rate it as bad, good or excellent. Based on voting, we sort and pick the top 3 voted masks and combine them.

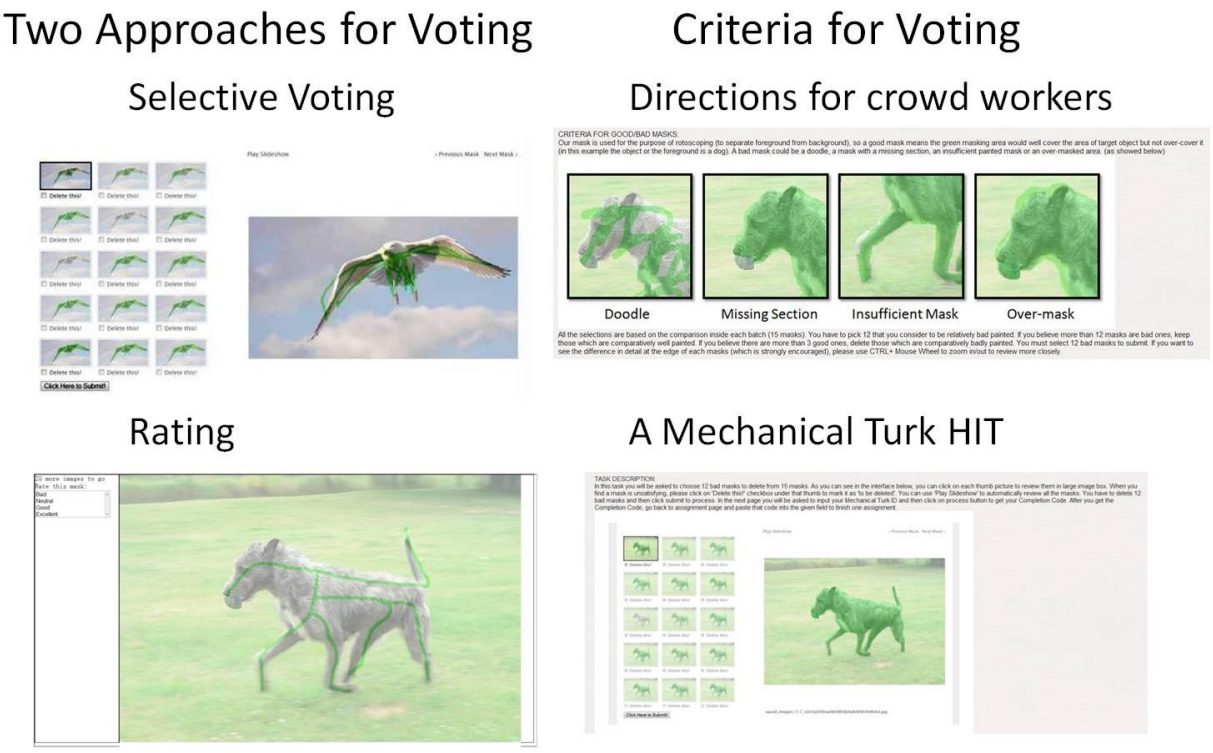


Figure 6: The voting approaches used, the criteria provided to crowd workers to assist in voting

We propose a simple combination algorithm for combining top 3 masks - for each pixel P in the image, we decide if it should be foreground or background by querying the same pixel on all the masks. If P is marked as a foreground pixel on more than N masks then we set P to foreground in the merged global mask, else as background. [Here N is heuristically chosen as 75% of all masks] Initially, one of the key design decisions for the pipeline was to have the ability to be able to refine a result by iterating over it over the crowdsourcing platform. After conducting experiments and user studies we found that little improvement in result led to an exponential increase in time and cost. Hence, we decided to pivot and focused on quick generation of result.

Implementation Details

The front end of the system which allows the worker to create masks and mark regions on an image is developed on Flash. The voting tasks and backend database are handled by PHP and MySQL. The scripts for mask pruning, combination and region generation are developed in Python (using the Python Image Library). The application development for the android platform is done using the Android SDK. For the user study, we deployed the application on Google Nexus One smartphone, made available to us by Prof Bjoern Hartmann.

Evaluation and User Study

This section covers the details of the system evaluations. We evaluate both the backend pipeline and the android application. The results of the evaluations are discussed in the results section.

Evaluation of the back-end pipeline

For the first few months, while developing a robust crowdsourcing back-end for our application, we conducted various experiments on Amazon Mechanical Turk to iterate over various components of our pipeline and develop a quick and cheap back-end while maintaining a minimum quality threshold. In the first experiment, we try to identify the need for combining work of varied users and need for iterative refinement. We compare results when (i) merged mask is the final output (as discussed above), (ii) only take 1 input mask at the beginning and iteratively refine the mask and (iii) merged mask is

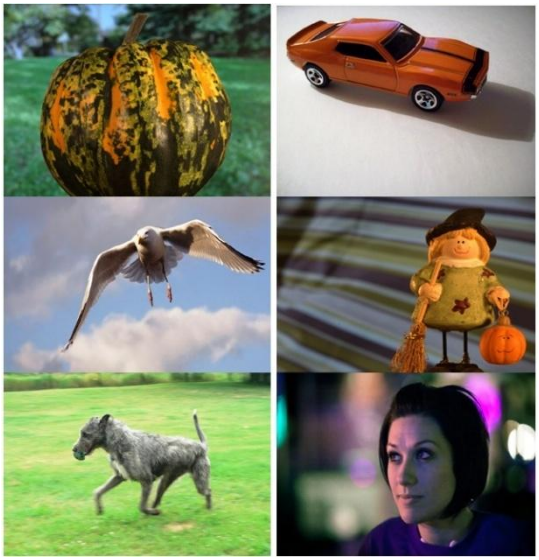


Figure 7: Test Images with complexity levels: Easy (1st Row), Medium (2nd Row) and Difficult (3rd Row)

iteratively refined. For this experiment, we take two images each from varied complexity levels: Easy, Medium and Difficult (Figure 7). We categorize an image based on two principal factors - foreground object's clear separation from background and level of detail in the foreground object. For example, pumpkin (easy) is clearly separable from background and has regular boundary, while the dog (difficult) has fur, so it lacks a clear boundary and moreover the boundary is irregular. We process each image 3 times through the pipeline. The results and findings are detailed in the results section.

Evaluation of the Android Application

In the user study, we put our Android Application in the hands of users and looked at the subjective experience of using the application as an end-to-end system for rotoscoping on mobile phones. We asked 20 users to use the application to capture an image with intent to change background later. We gave them minimal application navigation instructions and requested them to think out loud when using specific features in the application. The user study template can be found in the Appendix. We conducted two kinds of tests – one in which the user will get the result in less than 5 minutes (other team members worked in the back and generated a quick mask) and in the other, we let the mask come back from the crowdsourcing platform (which on an average took 3 hours). We made sure that the quality of the quick mask was comparable to the crowdsourcing solution. In our application, we provided users with 3 features which required varied level of user interaction with application to generate the final result. In "switch background", a user could select a new background image and place the mask anywhere on the background image by simply positioning it by touch interface. On the other hand, we also had a "sticker" feature where the user did not have much control on the output, and it required minimal user interaction with the application. We studied user interactions to determine the features which

interested users the most. We also studied user behavior patterns for features which required different level of interaction/ user inputs. This data consisted of information such as their familiarity with using Android smart phones, pattern of using new applications on the smart phones, pattern of spending money on applications (or in-application purchases), frequency of using photo-editing applications and also studied their sharing patterns on social media.

Results

This section summarizes results of the user study and comparison with Vision approach - grabcut

Results of experiments on Amazon Mechanical Turk:

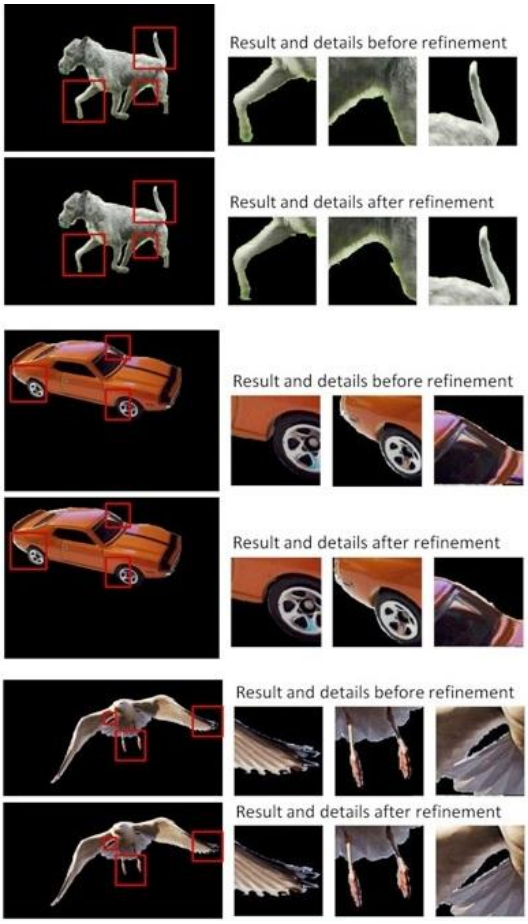
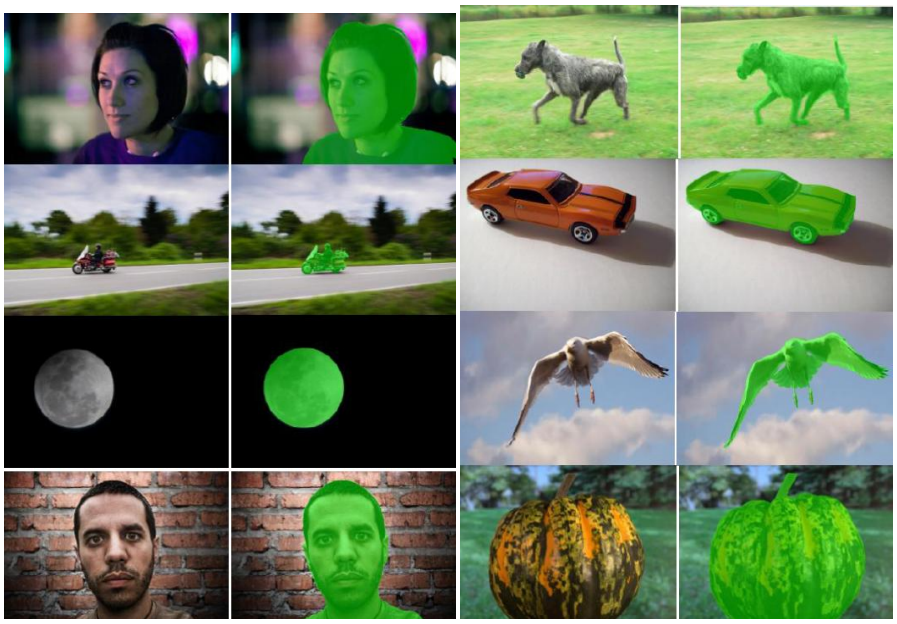


Figure 8: Iterative refinement on mask

Based on the studies conducted on the crowdsourcing platform for 10 images (each image 3 times), we find that regional refinement does not necessary create better result. It costs 5 times more than generating merged global mask and takes nearly 48 hours of time as compared to 3 hours for the latter. We also notice that there is no significant improvement in the quality of the mask for easy image, but we do see certain positive improvements in more complex images. (Figure 8) We also find that we cannot reliably generate high quality final mask, if we only take one input mask and iteratively refine it, as the foreground object correctly, and hence we do not have a good starting point to iterate on. Also, in this approach, as there is no guarantee that the

majority of foreground object will be marked by worker, we cannot perform iterative refinement based on threshold values. Finally, based on requirements of the final application, we preferred a quicker and cheaper solution– controlled combination. Results without iteration are shown below



For Images from top left to bottom right, Flickr user Ids of Image owner: (Left to Right)

Wolfhound, sebastian-krueger , 14043270@N08, Picfix, rosengrant , Mkamp, thomashawk, wolfhound

Figure 9: Final masks without iteration for Creative Commons Images (With Flickr user ids)

While running experiments on Mechanical Turk, we also tried to understand crowd worker behavior for a highly subjective task such as rotoscoping. We studied worker response time in relation to the complexity of the task (drawing v/s voting) and also examined the quality of work done by the workers - by examining the number of submissions rejected by the automated pruning script. After examining more than 400 submissions for the drawing task alone, we notice that 92% of the workers do a decent job at identifying and marking the foreground pixels. This number makes us believe that if subjective tasks can be explained well with strong visual graphics, we can indeed get useful work done on crowdsourcing platforms. Figure 9 shows the results of the experiments conducted to evaluate crowd worker performance on mask generation (without iterative refinement). Each image was run 3 times through the crowdsourcing platform, and 10 workers created masks for each drawing HIT, 7 workers helped in voting for each image.

Comparison with Computer Vision technique: Grabcut

We also compared the results from the crowdsourcing platform against the results from a popular Computer Vision technique: grabcut on the same images. We use this comparison to distinguish the images where computer vision techniques can develop a good segmentation from the ones where crowdsourcing solutions outperform the vision algorithms. The Grabcut algorithm models foreground and background color distribution by a Gaussian mixture model and models the image matting problem as binary classification of each pixel as foreground or background. We found that it works well for simple objects such as a cup, but as complexity increases, we see that crowdsourcing solutions perform better. We capture images using CrowdBrush application and compare results we get from the crowdsourcing system against the ones we generate from grabcut. For all the images, we also do a 2^{nd} iteration of grabcut and use it for comparison. The results can be seen in Figure 10. In general, we found that for all the images grabcut produced better boundary as compared to crowdsourcing solution, but in most images with humans, grabcut failed to recognize all the parts of the image which belonged to the human. In A, we see that there is no significant difference between crowdsourcing solution and the grabcut solution. In B, however, grabcut fails to recognize certain rings around the bottle, which semantically belong to the bottle and we see that over the crowdsourcing platform that information is retained and we get a semantically complete mask. Similarly in C and D, grabcut identifies some areas of surroundings (marked in Red) which do not belong to the human, while crowdsourcing solution performs much better. Lastly in E, there is no clear indication towards the right answer, but it is interesting to see that crowd workers converge towards a semantically correct solution. It is worth noting that grabcut provides high quality boundaries in sub-regions which are better than the crowdsourcing solution, but it fails to cover all regions which belong to the foreground object. Thus, we see that crowdsourcing provides more robust and consistent solutions.

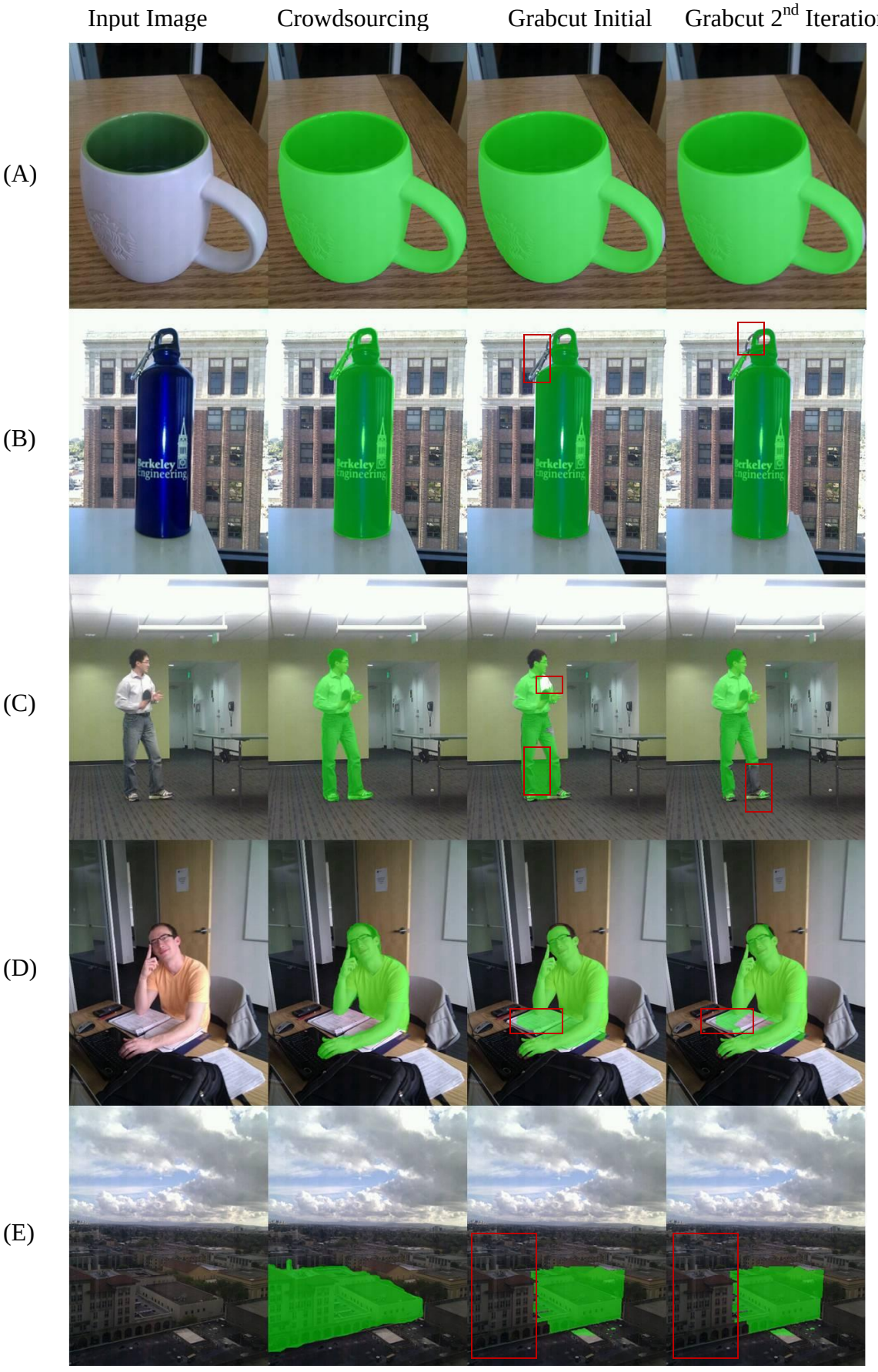


Figure 10: Comparison between Crowdsourcing and grabcut solutions. The red box shows errors in grabcut approach as compared with crowdsourcing solution.

Analysis of User study

In the user study 90% of the users were graduate students at UC Berkeley, belonged to the age group 18-25 and on an average had been using smartphones for 2 years. 60% of the users were iPhone users, while others used Android. 80% of the users used Facebook for sharing photos and on an average use social media on their phone 1-2 days every week. Only 10% users do in-application purchases and 20% said that they would be willing to pay 1.99$ onetime for this application. On a scale of -3 to +3, the average user experience rating was 1.5 and "switch background" was the most remembered feature of the application among the users. On an average, a mask costs us $0.40 and takes 3.5 hours on Amazon Mechanical Turk. We calculated this data for the 60 images we processed using the final back-end pipeline. A few of the final masks are shown in Figure 10. Figure 11 shows images generated using the CrowdBrush application with switch background (b), color splash (c) and sticker (d) features.

Figure 11: CrowdBrush image editing features. (a) Original Image (b) Switch background (c) Color Splash (d) Sticker

We found that users preferred a quick response compared to delayed response. We also found that users liked the idea of being able to interact with the mask and paste it on different backgrounds. This was interesting for us, as we initially hypothesized that users would prefer features which require minimum interaction. Around 75% of the users found the feature to be able to change backgrounds very interesting and engaging. 10 users actually used this feature the second time to be able to generate a better composite image. When asked if the feature was confusing for them, the general response was that they had never done anything like this before and thus after seeing the result of their action, they thought they could do better and hence used the feature the second time. Also, few users preferred if there was a way for them to edit the mask generated by the crowdsourcing system. The idea that they could have fun with their friend's image while on the go was of particular interest. Some users suggested that it would have been great if we provided more beach-like background images.

One feature which most of the users expected was the ability to scale up/scale down the mask. During the user study, I did notice a few users trying to scale up the mask. We would like to implement it in the next version of the application. Another common complain among the users was that the touch interface was not responsive enough. We expected that, as we felt it while developing the application. We think it is because the smartphone used by us – Google Nexus One is a few years old by now and the touch interfaces have improved greatly in the past few years. Some users also found the application to be slow and sometimes unresponsive.

Future Work

We believe that this work can be used to develop a pipeline to rotoscope a real-time video input. As video is not just a collection of independent images but a series of highly similar images

(frames), one can explore ways to exploit the temporal coherency of object position in a video. One technical difficulty will be to identify the key frames in a video feed. In this work, we did not experiment with providing smart drawing tools to crowd workers. One future extension would be to investigate if integrating state-of-the-art computer vision techniques into the drawing tool can increase worker productivity. We are aware of that these techniques have potential to increase a worker's throughput, but we also have concerns if these techniques will work for the crowd workers with diverse backgrounds. Currently we do not have an objective measure to determine how good a mask is. Availability of such a yardstick will help in better evaluation of the system.

Acknowledgements

I would like to thank Joel Brandt and Bjoern Hartmann for their continuous supervision throughout this work. Additionally, Joel Brandt provided us server access and funds to run experiments on Amazon Mechanical Turk and Bjoern Hartmann provided us Android handsets (Google Nexus One) to develop an android application and conduct user studies. I would like to acknowledge the creative common image owners, whose images were used for this study. Also, I would like to thank our fellow classmates at UC Berkeley who helped us in initial user testing of the CrowdBrush application and gave us invaluable insights and opinions about our UI design and application features.

References

[1] Kulkarni, A., Can, M. and Hartmann, B. Turkomatic: Automatic Recursive Task and Workflow Design for Mechanical Turk. CHI 2011.

[2] Dow, S., Kulkarni, A., Klemmer, S., Hartmann, B. Shepherding the Crowd Yields Better Work. CSCW 2011.

[3] Quinn, A., Bederson, B. Human Computation: A Survey and Taxonomy of a Growing Field. CHI 2011.

[4] Bernstein, M., Little, G., Miller, R.C., Hartmann, B., Ackerman, M., Karger, D.R., Crowell, D., and Panovich, K. Soylent: A Word Processor with a Crowd Inside. UIST 2010.

[5] Von Ahn, L., Dabbish, L. Designing games with a purpose. Communications of the ACM 51, 8 (Aug. 2008), p. 58-67.

[6] Little, G., Chilton, L., Goldman, M., Miller, R.C. Exploring Iterative and Parallel Human Computation Processes. HCOMP 2010.

[7] Spiro, I., Taylor, G., Williams, G., Bregler, C. Hands by hand: crowdsourced motion tracking for gesture annotation. CVPR Workshop on Automatic Vision with Humans in the Loop, 2010.

[8] Wang, J., Cohen, M. Image and Video Matting: A Survey. Foundations and Trends in Computer Graphics and Vision, Vol. 3, No.2, 2007.

[9] Chuang, Y., Curless, B., Salesin, D.H., Szeliski, R. A Bayesian Approach to Digital Matting. SIGGRAPH 2001

[10] Sun, J., Jia, J., Tang, C., Shum, H. Poisson Matting. SIGGRAPH 04.

[11] Wang, J., Cohen, M. Optimized color sampling for robust matting, in Proc. of IEEE CVPR, 2007.

[12] Bai, X., Sapiro, G. A geodesic framework for fast interactive image and video segmentation and matting, in Proc. of IEEE ICCV, 2007.

[13] Wang, J., Bhat, P., Colburn, A., Agrawala, M., Cohen, M. Interactive Video Cutout, in Proc. of ACM SIGGRAPH, 2005.

[14] Bai X., Wang J., Sapiro G.. Dynamic Color Flow: A Motion-Adaptive Color Model for Object Segmentation in Video. Proc. ECCV 2010.

[15] Bai X., Wang J., Simons D., Sapiro G. 2009. Video SnapCut: Robust Video Object Cutout Using Localized Classifiers. ACM Transactions on Graphics (Proc. SIGGRAPH), 28(3)

Appendix

User Study template:

Survey 1 (In - Application testing)

1. Do the following:
   Login, Take picture, Save picture, Submit the picture for mask generation, wait till mask comes back, change background.
2. Select the mask,try the sticker feature, save the image.
3. Select the mask,try the color splash feature, save the image.

Survey 2
1. Sex : Male / Female / Prefer not to answer
2. Age group: 18-25 / 26-35 / 36+ / Prefer not to answer

3. Education background: High school / College Degree / Masters / PhD / Prefer not to answer
4. How long have you been using smartphones? ____ years and ____ months
5. Have you used both Android and iPhone? _____
6. Which one are you using currently? _____
7. How often do you use camera in your smartphone to take pictures?
Less than once a week /1-2 times a week/ Once a day/ More than 3 times a day
8. Do you use social media on your phone (ex: facebook, google+, twitter, tumblr etc)? Yes/ No
If yes, which ones do you frequently use?
_____
9. Do you upload pictures to the social media websites from your phone? Yes / No
If yes, which ones (check all that apply) - facebook / twitter / tumblr / google+, __
10. How often do you upload to social media websites using your phone?
Less than once a week /1-2 times a week/ Once a day/ More than 3 times a day
11. Which photo uploading applications do you use, if any?
_____
12. Any specific features you like about that application/ What made you choose that application?
_____
13. Do you buy apps, or spend money on in-app purchases? Yes/No
If yes, which apps do you spend money on? _____

Survey 3 (Post Application testing)
1. How was the experience? (-3: Hated it, 3: Loved it)
-3 -2 -1 0 1 2 3
2. Did you feel that the application did something unexpected during your navigation? Yes/No
If yes, explain _____
3. Do you think if something like this was available for free, you would use it? Yes/No
4. If you were to pay for this application, how much would you be willing to pay? _____
5. Please list the features (which you remember) this application has.
_____
6. What feature interested you the most, why?
_____
7. Is there something else you would have wanted to do with the mask?

---

8. How much did the amount of wait time influence your likability of the application?
-3 -2 -1 0 1 2 3
9. What kind of images do you think you would like to use this application for?
_____
10. Anything else you would like to add /comment?
_____