Doubleshot: An Interactive User-aided Segmentation Tool

Tom Yeh
CSAIL, MIT
Cambridge, MA
USA
1-617-253-4278
tomyeh@mit.edu

Trevor Darrell
CSAIL, MIT
Cambridge, MA
USA
1-617-253-8966
trevor@mit.edu

ABSTRACT
In this paper, we describe an intelligent user interface designed for camera phones to allow mobile users to specify the object of interest in the scene simply by taking two pictures: one with the object and one without the object. By comparing these two images, the system can reliably extract the visual appearance of the object, which can be useful to a wide-range of applications such as content-based image retrieval and object recognition.

Categories and Subject Descriptors
H.5.2 [**User Interfaces**]: *Input devices and strategies,* I.4.6 Segmentation.

General Terms
Human Factors

Keywords
Object recognition, computer vision, mobile application

1. INTRODUCTION
The popularity of camera phones has opened up a new avenue for image-based mobile applications thanks to the extra input modality the built-in camera can offer besides traditional ones based on keypad, voice, and touch-screen. For example, KDDI has been providing services for its subscribers to retrieve information based on special visual markers printed on anything ranging from business cards to telephone poles [2]. Also in [4], a landmark image of the current location can be used as the query to search the web for useful information by finding other web images containing the same landmark.

Perhaps the application that has generated the most excitement in both academia and industry is object recognition on camera phones. But one difficult issue needs to be solved: how to distinguish which part of the input image the object actually occupies? And if there is more than one object in the scene, which one interests the user? This issue is known in the field of machine vision as "image segmentation." However, an automatic and reliable solution has remained an open problem in the field.

In this paper, we propose an interactive solution and show it is possible to exploit human abilities through an intelligent interface to make image segmentation tractable with existing technologies.

IUI'05, JANUARY 9–12, 2005, SAN DIEGO, CALIFORNIA, USA.

2. INTERACTIVE USER-AIDED SEGMENTATION
The idea of using human to guide segmentation has been explored before [3]. But they require the users to operate on the desktop environment working on a stationary image with a pointer device such as a mouse. This is not desirable since the object recognition system is most useful in a mobile setting: mobile users would like to receive just-in-time information as soon as they become interested in am object rather than waiting for the next opportunity to upload its images to a desktop computer.

What we want is an interactive human-aided segmentation tool where human input is obtained online when the image of object is being taken with the camera, which implies a light-weight algorithm that can be implemented on typical camera phones. The obtained human input can be sent to a powerful server for further processing. No longer constrained by the computing resources, the server can carry out more sophisticated algorithms. Hence, the goal of the intelligent interface on the client-side is to collect the most amount of user input to provide the server-side with adequate information to perform reliable image segmentation.

However, in designing an interface, care needs to be given to the amount of cognitive and physical effort that is required of the user. If the interaction is too tedious, the overall usability can be greatly compromised even though it may achieve perfect segmentation results. For example, although it is possible to ask the user to meticulously draw the contour of the desired object directly on the device with the keypad, it can be very tiresome when it comes to complex-shape objects.

3. DOUBLE-SHOT APPROACH
The interactive segmentation tool we propose will require the users to perform a simple task of taking only two images: one with the object and one without the object (see Figure 1). We call it *Doubleshot* to reflect this. The basic idea here is to treat the object as the difference between two images. We will first describe how the client-side interface facilitates the capturing of the two images and then explain how the server can obtain segmented object image from the two images.

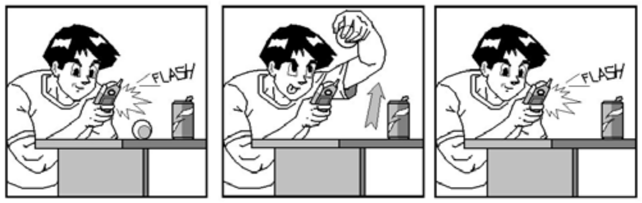

Figure 1: DoubleShot image segmentation. A user specifies the object of interest (ball) by moving it.

3.1 Client-Side Interface

The client side interface can be broken down into three steps each of which is meant to specify some aspect of the object identity.

3.1.1 Where is the object located?

First, the user simply aims the camera at the object and takes an image. We can at least have the confidence that the object must appear somewhere in the image (see Figure 2(1)). In addition, a dashed rectangle box is drawn in the viewfinder to encourage the user to center the target object, which can be very helpful later for the server to segment the object image.

3.1.2 Which object in the field of view is the user interested in?

Since the object can be situated in any background with an arbitrary number of other objects in the scene, it is necessary to determine which object exactly the user cares about. Here, the user can indicate her intention by grabbing the object and removing it from the scene. After the object is removed from the original background, the user can take a second image at the same camera angle and position. Then, the object that has disappeared from the second image must be the very object that interests the user.

The segmentation can be done simply by comparing each pixel's color values in the two images, an approach which is feasible with the limited computing resources on a camera phone. Unfortunately, during our preliminary experiment with this approach, we discovered that it is very tricky to hold the camera still due to respiration, the action of object removal, and unsteady hands. Therefore the two images are very rarely aligned. A pixel in one image does not necessary correspond to another pixel at the same location on the other image.

Instead of requiring the user to hold the camera exactly fixed we instruct the user to move the camera but provide an interactive segmentation cue. As the user moves the camera around, the image in the viewfinder is constantly changing to display what is being captured, but with a semi-transparent overlay of a grayscale copy of the first image. The user can match the moving image of the current scene to this stationary image (see Figure 2(2)). Once the two images are aligned, the user will have regained the original camera position of the first image. This is not difficult—for example, to make the image larger, one can simply move the camera closer. Such intuition of how the viewfinder image changes to reflect the camera's motion is a reasonable camera skill that can be expected from someone who would own a camera phone.

To further assist the user during the camera-alignment process, small red-dots are drawn over the area of the viewfinder image that has different color values from the first image. This area is what the system would consider the object if the second image were to be taken at this point. In effect, it provides the user with an additional visual cue on how well the two images are aligned. A good alignment will cause all of the red-dots to fall exclusively in the object area, with few red dots in the background area. Once the user manages to restrict the red dots to the object area, the user will know that "it's time to snap.'

The rendering of the red dots does not require complex computation. For each pixel in the first image, we check if there is a similar pixel in the 3 by 3 neighbor around the corresponding location in the second image. If one exists, it means this pixel is a background pixel and a red-dot is drawn. The similarity is measured as the square distance in the RGB color space, which can be done using only integer arithmetic.

3.1.3 How distinct is this object from the background?

In an ideal world, the background pixels are the pixels that retain the same color values from the first image to the second image. However, due to camera noise this is often not the case. We can only postulate the background pixels are those "less" different from the others, and the object image is those pixels that are "more" different from the others. Asking how much is "less" or "more" is equivalent to asking how distinct the object is from the background---how much is "stands out". We delegate to the user the judgment of picking the right level of tolerance to dictate to the system how much it should tolerate the difference for background pixels. The system uses this tolerance measure to set a threshold on how large a difference in pixel intensity between the two images must be for the pixel to be considered part of the object (see Figure 2(3)).

3.2 Server-Side Processing

Since we are no longer restricted by the computation resource, we

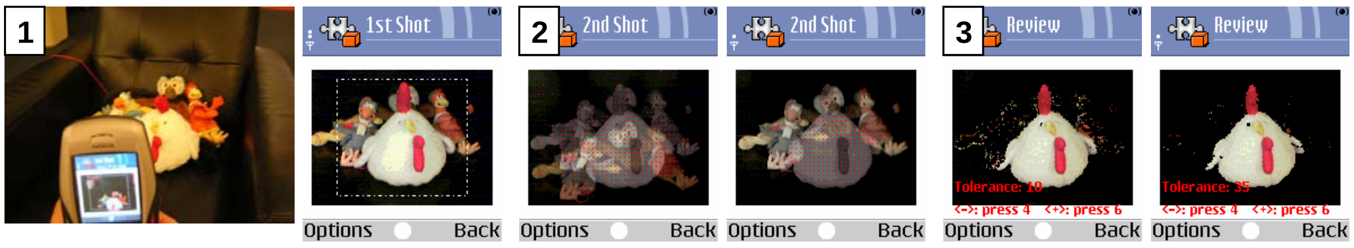

Figure 2: Client-side interface: (1) take an image, (2) align the second image with the first, (3) pick a good tolerance value.

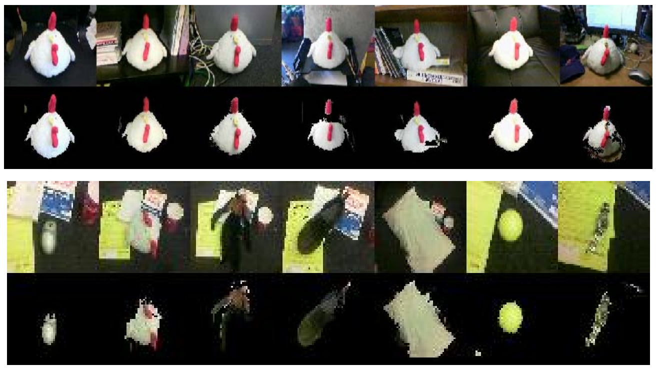

Figure 3: Sample segmentation results. (Top) Same objects with different backgrounds. (Bottom) Different objects with the same background.

are free to adopt more advanced computer vision techniques. We can compute the projective transformation matrix between the two images using an iterative gradient descent algorithm similar to [1] with the objective function defined as the correlation between the background areas in the two images. The best alignment occurs when the best background correspondence can be established between the two images. Thanks to the client-side user interface, the two images are fairly aligned, if not perfectly. At worst, they are off by a few degrees of rotation and few pixels of translation and scale. The search space is small and the algorithm can often converge within a few iterations. Then, we can determine the object area by comparing the two well aligned images in a similar way as the client-side. Assuming the object consists of only one part, we keep the largest connected region around the center of the image. We smooth the edges and flood-fill the area within the object contour for patching up small holes to obtain the final segmented object image.

Recall that when the image was being taken, a rectangle box is drawn on the viewfinder. A soft constraint is imposed on the user to fit the object in the box as much as possible. This provides a ground to adopt a probabilistic model regarding the object area—a pixel is more likely to be a background pixel if it falls outside the box than inside the box. Weights can be assigned accordingly when computing the correlation between the background areas.

4. IMPLEMENTATION

We implemented the *Doubleshot* segmentation tool on a Nokia 6600 device with a built-in camera, 64K colors, and a 640 by 480 resolution. The software was written and run on the native Symbian operating system. The actual image size processed by our system is 160 by 120. We were able to run our interactive method on the phone at the rate of at least 25 fps. On the server side, the segmentation routine takes less than 0.1 sec. in Matlab on a Pentium 1.5GHz PC.

5. RESULT AND DISCUSSION

We have tested the interactive segmentation tool on a wide range of objects with various backgrounds. Figure 3 shows the results of segmenting the same object (i.e., a toy animal) against different backgrounds. The same figure also shows the results of segmenting different objects against the same background. In most cases the image of the object was successfully segmented. This demonstrated the overall effectiveness of the tool.

Some segmentation errors did occur. For instance, in the last example with the toy animal, the object cast a strong shadow onto the background which would look different (i.e., brighter) after the toy was removed, causing the shadowed region to be mistaken as part of the object. Also, in an unfortunate event when the background happened to look similar and act as a camouflage for the object, the segmentation quality suffered as in the 4th and 5th toy animal example. However, we believe as users become more familiar with the interactive tool, they will learn to recognize such vulnerability and put the object against a more suitable background to apply the segmentation tool.

6. CONCLUSION AND FUTURE WORKS

We have described an intelligent interactive interface that allows users to specify the object of interest in the environment. By taking two images, the visual appearance of the object can be reliably obtained. In the future, we would like to extend the idea of interactive user-aided segmentation to immovable objects such as a statute in the museum; one possibility is to synthesize the object image from a set of images taken from various viewpoints by considering their 3D correspondence. Also, we hope to tie our interactive tool to an image matching module and build a functional image-based object recognition system for mobile users.

7. REFERENCES

1. Lucas B. and Kanade T., An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 674-679, 1981.
2. New KDDI CDMA wireless handset 2D codes, http://www.3g.co.ul/PR/Jan2004/6278.htm.
3. Rother, C., Kolmogorov, V., and Blake, A., GrabCut: interactive foreground extraction using iterated graph cuts, *ACM Transactions on Graphics*, Vol 23, 309-314.
4. Tollmar, K., Yeh,T., and Darrell, T., IDeixis: searching the web with mobile images for location-based information, *Mobile HCI 2004*: 288-299