"Dancing Icons" Detection

Itamar Friedman
Technion
Haifa, Israel
ItamarF@tx.technion.ac.il

Lihi Zelnik-Manor
Technion
Haifa, Israel
Lihi@ee.technion.ac.il

Abstract

Undoubtedly, a key feature in the popularity of smartphones is the numerous applications one can install. Frequently, we learn about applications we desire by seeing them on someone else's mobile device. A user-friendly way to obtain these particular applications could be by taking a photo of their corresponding icons as displayed on our friend's screen. We then need to develop methods for automatic detection and recognition of the icons in a screen shot. This paper suggest a method for icon detection (icon recognition is left for future work).

The variety of icons (∼500K) and wallpapers makes the detection task very difficult using methods such as edge detection, contour detection or template matching. In order to bypass this difficulty, we suggest using a special feature introduced in several smart phone. When one enters the edit mode for organizing icons on the screen, the icons vibrate. Furthermore, When one moves from one set of icons' view to another, the icons slide on the screen. We use this feature to obtain a better detection of the icons on the screen.

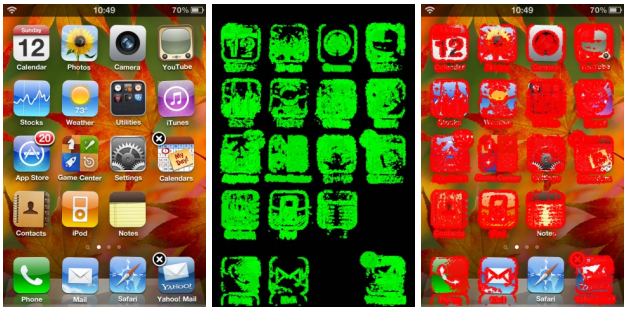
1. Introduction

The solution we propose enables the detection of multiple icons by taking two snapshots of the screen, while in the "vibrate" mode. Our approach to detect the icons' frames is by finding their contours in the optical flow image. The optical flow image is produced from two aligned snapshots that contain "Dancing Icons". From the flow image we would want to understand where are the potential points that can reveal the icons' locations. By projecting those points on a blank image we find potential contours. We will use a-priori information in order to find contours that can describe icons' frames. Finally we use local and global structures and information in order to determine the final detection from the acquired contours.

2. Proposed Approach

2.1. General

In the "vibrate" mode the background is stationary while the icons rotate slightly. Hence, the optical flow between a pair of aligned snapshots should be close to zero on background pixels, and significant on icon pixels. This is demonstrated well in Figure 1, which shows the magnitude of the optical flow vectors computed for a pair of aligned screen shots. This suggests that computing the magnitude of the optical flow at each pixels could provide useful information.



(a) (b) (c)

Figure 1. The optical flow magnitude between a pair of aligned snapshots is large on icon pixels and minimal on the background. (a) Frame 1 (b) Optical Flow magnitude (c) Overlayed on Frame 2

2.2. Algorithm

Capturing two frames and aligning them: Our algorithm expects as input a pair of images taken during the "vibrate" mode. In case they are not aligned, we align the image by fitting a homography to feature matches using RANSAC. We then warp one of the images to obtain alignment. This is illustrated in Figure 2.

Computing Optical Flow: We chose to experiment with three algorithms for optical flow estimation: the parametric approach of Ferneback, the seminal algorithm of Lucas-Kanade and our home-brewed algorithm based on Block

Figure 2. To align a pair of frames we fit a Homography using RANSAC and warp one of the frames.

Matching. The result of Lucas-Kanade and Fernback are very noisy, while the result of Block-Matching is significantly cleaner, with practically no false motion vectors detected on the background.

Icon contour detection: We used two types of edge detection: Canny and simple thresholding of the optical-flow magnitude image. We applied a set of thresholds to obtain multiple edge detection results. Each type of "edge image" reveals different contours.

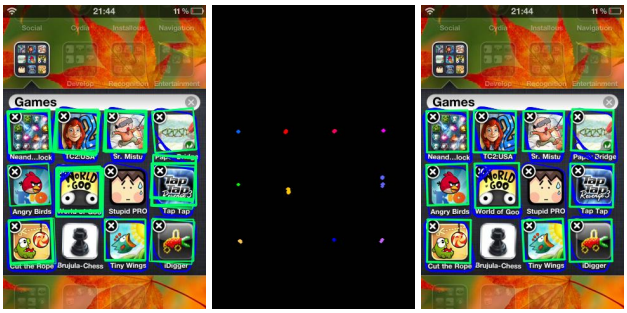
We adopt the following procedure to detect candidate contours. We first use Suzuki's contour detector while allowing it to connect nearby contours into one chain. Next, we approximate the contour with a low dimension polygon curve using Douglas-Peucker algorithm. We then exclude contours whose approximated polygon doesn't fit our prior assumptions of icons. We create a convex contour from the original contour using convex hull algorithm of Sklansky. This yields the blue contours shown in Figure 3. Finally, we extract a square that best fit an icon in the contour, while filtering squares that do not fit our prior assumptions. Our result is illustrated by the green squares in Figure 3. It can be seen that almost every icon is surrounded by several contours, and several squares that passed the entire filtering process.

(a) (b) (c)

Figure 3. (a) Contours detects by our system. (b) Candidate bounding-boxes matched to the contours. (c) The final detected boxes

Finding a bounding-box for each icon: Our final step consists of grouping candidate squares into our final icon detection. Our approach is based on the k-means algorithm, however, we wish to by-pass the need for setting the number of groups k manually, as we do not know the number of icons.

The common method for detecting k automatically is superfluous and time consuming. Our approach is based on transforming the problem into the image domain. We mark the center of each candidate box on the image plane, see Figure 4. We then apply a Suzuki's contour detector to this image. Each group of points is surrounded by a single contour. All boxes whose centers were surrounded by the same contour are grouped together. The final box for each group of icons is defined as the mean over all corresponding candidates.



(a) (b) (c)

Figure 4. Grouping box candidates. (a) An input image with the candidate boxes marked in green. (b) The centers of the candidate boxes. We apply contour detection to this image and hence discover the number of centers automatically. (c) Boxes detected after grouping. A single box per icon is obtained.

Refining using global arrangement: The local properties of each group are noisy and hence often the resulting boxes of the previous stage are inaccurate. To further reduce these inaccuracies we use global properties known regarding the positioning and size of icons. These global properties are determind by (1) a-prior known global information and (2) global information acquired from the current pair frames.

3. Conclusion

In this paper we have proposed a practical algorithm for automatic detection of icons in snapshots of a smart phone, taken during the "vibrate" mode. It then computes the motion between the snapshots and uses the motion vectors to detect the icons. Our experiments suggest a high detection rate (91.5%) with no false alarms. Further improvements can be obtained by further introducing global constraints and by detecting the icon sub-types, i.e., some icons are attached flags which modify their rectangular shape.