Rectangular Target Extraction for Mobile Augmented Reality Applications

Andreas Hartl and Gerhard Reitmayr
Graz University of Technology, Austria
{hartl, reitmayr}@icg.tugraz.at

Abstract

We propose a fast and efficient method for localization and rectification of a dominant rectangular region within an image, particularly suitable for mobile Augmented Reality applications. This approach can deal with perspective distortion and high-frequency structures such as text. The resulting image may be used for planar tracking or as input for subsequent image processing tasks. We demonstrate the feasibility of our approach in an extensive evaluation on off-the-shelf mobile hardware. We show its application in two particular use cases, namely planar tracking and mobile visual search. In both cases significant improvements in performance are achieved by using rectified images of the region of interest.

1. Introduction

Planar structures occur frequently in our environment. Among them, there is a large subset having rectangular borders. Examples are all kinds of printed documents, posters, a deck of cards, the screen of a computer, a window or an image projected onto a wall. This ubiquity makes them an interesting target for mobile Augmented Reality (AR) applications.

Mobile phones have become a popular platform for AR due to advancements in processing power, camera quality and algorithms [16]. A large group of mobile AR applications currently employ natural feature tracking (NFT) to obtain the pose of a target in real-time. They typically use one or more rectangular image targets which are delivered with the application. Consequently, the targets cannot represent the effects of operating conditions and camera settings on appearance occurring at runtime.

Specific applications may require to track a variant of a known target. This may degrade robustness due to a lower number of matches. In case of a larger number of variants, deploying them with the application may not be feasible. Even then, tracking is still limited to a known list of targets.

Being able to instantly create a tracking target enables more interactive applications and improves the performance of established ones. More specifically, an extracted target can be used for subsequent image processing tasks instead of the full frame image.

Rectangular regions of arbitrary content and size may be localized by using variants of the Hough transform [6, 1]. In general, these have high computational demands or cannot deal with perspective distortion. This can be overcome by taking into account vanishing points or lines [7, 14, 10]. Success depends on the availability of this information or the process may again not be feasible on mobile devices. Then, there are methods employing edge and line primitives [15, 9], possibly generating and verifying a list of hypotheses. Still, these may be susceptible to image noise or lack perspective invariance. A more general approach is the application of a planar SLAM technique in a mobile scenario [12]. In this case, feature points need to be computed and no exact borders of the plane are given.
We propose a method for localization and rectification of a dominant rectangular region within an arbitrary image. This approach can deal with perspective distortion and high-frequency structures (see Figure 1). We show the success of the approach as well as its applicability to mobile devices in an extensive evaluation on real hardware (see Section 3.1). We obtain a significant increase in tracking performance (see Section 3.2), and are able to substantially improve the recognition rate of an exemplary visual search application by processing a rectified image obtained from the detected rectangular region (see Section 3.3).

2. Localization of Rectangular Structures

Rectangle detection using the Hough transform is not robust w.r.t. perspective distortion and harms efficiency due to the high dimensionality of the accumulator space. We also noticed that vanishing point estimation often fails on typical images showing rectangular tracking targets. Using image primitives resolves these issues. In contrast to [15] or [9], we employ adaptive edge detection, filter high frequency noise and can deal with a reasonable amount of perspective distortion.

To ensure broad applicability, assumptions on the content of the background image as well as the region to be extracted should be avoided. However, we limit ourselves to planar rectangular regions that are visually separated from the background through their bounding edges, lying within a region of interest (ROI). This is a trade-off to obtain an algorithm that is computationally feasible on mobile platforms and to improve robustness.

2.1. Algorithm

Initially we compute an edge map by applying a Canny edge detector [3] with automatic threshold selection (see Figure 2 for an illustration of the steps involved). After filtering text-like structures, we perform line detection to obtain all possible lines within the ROI [5]. We group each two lines having a difference in direction of less than 14 degrees together. Then, pairs of line bundles are selected, giving a list of hypotheses for rectangular regions, each consisting of four lines. In order to reduce the amount of initial hypotheses, filtering is applied by intersection of lines and verification that the corners lie within the ROI. This is a critical step that improves robustness as well as efficiency. Finally, the support for each hypothesis is computed on a dilated edge image, which forms the basis for ranking. This is to account for imperfect fitting and a certain amount of curvature. The result is a ranked list of hypotheses, from which the final candidate is taken. This corresponds to the most dominant rectangular region w.r.t. the ROI.

If the region is to be used as a tracking target, we assume it to be rectangular, so that it can be automatically rectified. We compute the dimensions of an undistorted rectangle by averaging the pixel width and height of the corresponding hypothesis. With this information, a homography can be computed for rectification.

2.2. Text Filtering

High frequency structures can lead to rather strong, but false responses in line detection. Consequently we perform filtering of the edge image before line detection (see Figure 3). This improves robustness as well as runtime of our algorithm by reducing the amount of hypotheses. In this step, we compute connected components [4] from the adaptively thresholded [13] input image and assess them according to the criteria aspect ratio, relative height and amount of pixels with respect to their bounding boxes. We compute each criterion w.r.t. the dimensions of the ROI.

2.3. Adaptations for Mobile Phones

As the scenario for image acquisition is largely unconstrained, an automatic selection of parameters is desirable, especially for edge detection (see Figure 3). However, the flood fill operation used for hysteresis thresholding in the Canny edge detector was highly inefficient when run on mobile devices. Switching to a stack-based implementation gave a more than 100-fold speed-up compared with a standard implementation. In addition, line detection was a major bottleneck. We achieved a 4-fold speedup by using look-up tables and...
reducing the resolutions of the accumulator space. This allows instant processing on current mobile hardware.

3. Experimental Evaluation

We first assess the general performance of our algorithm concerning accuracy and runtime. Then, we investigate the performance of the extracted target in NFT. Lastly, we demonstrate that the approach can be beneficial in a visual search scenario. We use the Samsung Galaxy S2 smartphone for all relevant parts of this evaluation (ARM A9-based dual-core CPU up to 1.2 GHz, 1 GB of RAM, 8 MP camera). We developed two mobile prototypes using the proposed extraction approach in a mobile NFT pipeline demonstrating exemplary application scenarios (see Figure 1).

3.1. Accuracy and Runtime

We took 78 images (640 x 480 pixels) of various categories of rectangular items (book covers, business cards, posters) and manually annotated the rectangle corners. For lower resolution images, we downscaled the ground truth locations with the images.

For assessing the quality of localization, we compute correspondences between the extracted and manually annotated corners and the distances between corresponding corners. We define a relative error metric,

\[ e = \frac{\Delta d_{ex_{max}}}{d_{ref_{min}}} \]

where \( \Delta d_{ex_{max}} \) denotes the maximum error in location over four corners and \( d_{ref_{min}} \) denotes the minimum edge length for the annotated quadrangle. According to our results the relative error decreases w.r.t. the input resolution (see Figure 4).

In practice, we found \( e = 0.035 \) to be a suitable upper threshold indicating successful detection of a target that is usable for all kinds of applications. Nevertheless, the target can be used for AR applications even with a larger relative error.

3.2. Effect on Natural Feature Tracking

We compared tracking performance of a digital representation with that of an extracted target in three conditions: indoor, outdoor and a very low light environment. For this purpose, we implemented a NFT pipeline on mobile phones. We use BRISK [8] as a detector/descriptor in the computation of the initial pose and hand over this information to a patch-based tracker [16].

We process 60 video sequences (640 x 480 pixels) of posters and books and determine the percentage of successfully tracked frames (track-ratio) with respect to all frames when using the digital image or a target image extracted from the video (see Figure 5). We consider tracking to be successful if at least \( th_{kp} \) percent of all visible keypoints match w.r.t. a given NCC-threshold.

<table>
<thead>
<tr>
<th>Res.</th>
<th>Edges</th>
<th>T-filter</th>
<th>Lines</th>
<th>Hyp.</th>
<th>H-filter</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>320x240</td>
<td>62</td>
<td>23</td>
<td>37</td>
<td>69</td>
<td>24</td>
<td>4.7</td>
</tr>
<tr>
<td>480x360</td>
<td>151</td>
<td>63</td>
<td>84</td>
<td>93</td>
<td>7</td>
<td>2.9</td>
</tr>
<tr>
<td>640x480</td>
<td>264</td>
<td>111</td>
<td>129</td>
<td>28</td>
<td>3</td>
<td>1.9</td>
</tr>
</tbody>
</table>

Table 1: Mean runtime in milliseconds for all relevant parts of our algorithm including overall framerate (Samsung Galaxy S2)

We evaluated the runtime performance of our algorithm on a mobile device. Table 1 shows the individual processing steps and overall processing time for different input resolutions. Note, that the time consumed for the hypothesis-filtering step at 320 x 240 pixels is around twice the time required by the two higher resolutions. This can be attributed to merging of high frequency structures which cannot be filtered by our region-based text filtering. Overall, we chose an input resolution of 480 x 360 pixels for the remaining experiments as it represents a good tradeoff between accuracy and runtime. In specific cases the lowest resolution still performs adequately which allows to have near-interactive frame rates on the tested device.

Figure 4: Mean relative error per category w.r.t. input resolution

Figure 5: Left: tracking performance w.r.t. environment for digital and extracted targets; Right: Recognition rates for visual search using full image and extracted region
th\textsubscript{NCC}. We use \(th\textsubscript{kp} = 10\) percent and \(th\textsubscript{NCC} = 0.68\).

In all situations, the extracted target gives a higher ratio of tracked frames. According to our experiments, the expected gain can be more than 25%, depending on the environment of operation.

3.3. Improving Visual Search

In this experiment we compare the recognition performance between the full input frame and the extracted and rectified image for books, business cards and posters. We implemented a vocabulary tree-based visual search approach [11] using SURF [2] features.

We are able to improve the performance by 8-25% depending on the object category (see Figure 5). The business card category is particularly difficult for traditional recognition, because it contains several examples having the same layout but with different personalization. In this case processing the extracted and rectified image gives a considerable gain in recognition rate.

4. Conclusion

We presented an efficient approach for localizing a dominant rectangular region within an arbitrary input image suitable for current off-the-shelf smartphones. We demonstrated the applicability of the approach in an extensive evaluation on real embedded hardware with several object categories under realistic operating conditions. According to our results, tracking performance improves by more than 25% when using the extracted target. When performing visual search on the rectified image, a performance increase of around 10% is feasible. Difficult categories such as individual business cards improve by 25%.

The approach could be improved in several ways, which we are investigating for future work. The procedure may fail if the scene contains grid-like structures (see Figure 6). To resolve these situations, we plan to include user feedback. Tuning performance to operate at frame-rate allows instant feedback and letting the user choose a good frame. Tracking the resulting target over several frames could then be used to filter out erroneous responses. Curved contours may hamper detection. We currently address this problem by allowing a certain amount of deviation from a straight line. A perpendicular edge search could obtain a better estimate of the contour. While possibly making the process more accurate or successful, the effect on runtime needs to be investigated.

Acknowledgements: This work is supported by Bundesdruckerei GmbH.

References