Machine Perception Assignment 2  
House Number Extraction

Reece Jones

Abstract

This report presents a pipeline for extracting and recognising house numbers from constrained real-world images. The pipeline utilises maximally stable extremal regions (MSER), hand-crafted region filtering and text path extraction logic, and support vector machine (SVM) digit classification. Perfect accuracy is achieved on the provided training and validation images.

1. Introduction

The problem of extracting text from real-world images is difficult, and likely remains an open problem. Several existing methods were examined, [1], [2], [3], and [4], all of which utilise large-scale machine learning in some manner to deal with the variations present in natural scenes (lighting, affine transformations, perspective, etc.). A particularly difficult element of this problem is the extraction of the characters, typically requiring machine learning to classify character and noncharacter regions. Fortunately, as large-scale machine learning is not feasible for this task, the images provided are constrained enough such that hand-crafted algorithms can achieve good performance. Primarily:

* Digit fonts are simple and limited,
* Digits are mostly uniform, bright colours on uniform, dark backgrounds,
* Numbers are near to horizontal,
* Limited perspective warping is present,
* Images contain at most one house number each, and
* Any other text present is small.

Inspiration was taken from [1], as the methodology was similar to my own thoughts on the task and could be adjusted to require less machine learning.

1. Pipeline Methodology

The developed pipeline consists of three main sections: digit region detection, region filtering, house number extraction, and digit classification.

A. Digit Region Detection

Multiple techniques exist to detect regions in which text may be present. This is not necessarily a trivial problem. A simple threshold or Otsu’s binarisation is unlikely to be robust, as no guarantees can be made about the nondigit regions of the image – these approaches are too global. Adaptive thresholding, image gradients, edges, and contours were considered, but the results were difficult to utilise and were not reliable. One solution used by [2] and [3] is a sliding-window approach, where the entire image is mapped to a surface which represents the likelihood of text being present. However, this approach requires evaluation of arbitrary image patches for containment or noncontainment of text, likely requiring a machine learning model for which we do not have sufficient data nor computational resources. Attempts without machine learning (i.e. hand-crafted features) were too slow in Python to be feasible. After running out of ideas, the approach from [1] was used: assume each digit is a connected extremal region and use MSER blob detection. MSER is computationally efficient and succeeds in detecting the digits in all the provided images.

1. Region Filtering and Adjustment

MSER detects many regions which are not digits, so some manner of filtering is required to remove nondigit regions. An initial idea was to use the digit classifier to ignore regions that were not confidently classified, but SVMs do not really support this functionality. Another approach, used by [1], is to train a separate classifier for discriminating between digit and nondigit regions, but is, as explained in the previous paragraph, not feasible for this task. Therefore, a series of hand-crafted filters is implemented, based on observations of the provided images.

The first filter is based on regions’ bounding box areas. Any regions whose bounding box area is less than 50 pixels in area is discarded, deemed too small to contain a recognisable digit. Any regions whose bounding box is greater than 25% of the total image are is discarded, as the digits in the images provided are typically not this large. This

The second filter is based on the regions’ bounding box aspect ratios. Digits are typically taller than they are wide, so if the aspect ratio is greater than 0.8, the region is discarded. Digits are typically not extremely thin, though, so if the aspect ratio is less than 0.2, the region is discarded also. This filter is a simple and efficient method for removing a large proportion of undesired detected regions.

The third filter considers the pixel intensity of the regions. One of the constraints on the provided images is that the digits are a light colour on a dark background, i.e. the digits are bright compared to their local area (this fact is also relied on for processing further in the pipeline). However, MSER produces regions which are both locally bright and dark. To filter out the locally dark regions, a region’s pixels are compared to the foreground pixels determined by Otsu’s binarisation of the image patch within the region bounding box. If less than 75% of the region’s pixels are foreground, the region is discarded.

The fourth filter considers the connected components within the regions’ bounding boxes. Digits consist of a single foreground component, but MSER may yield regions whose bounding boxes span multiple foreground components. To filter such regions out, the image patch within the region bounding box is binarised with Otsu’s binarisation, the connected components are found, and the areas of the components examined. If there is more than one component with an area greater than 5% of the region bounding box, the region is discarded.

The fifth filter also examines the connected components within the regions’ bounding boxes. Digits are typically not connected to any other foreground regions, yet MSER’s regions are sometimes subregions of the local foreground. This filter attempts to expand the bounding box of a region by a few pixels until the border pixels do not contain any of the largest foreground component. If this can be done, the original region’s box is replaced with the expanded box. If this cannot be done, the region is discarded.

As a final step, duplicate regions are removed. MSER and/or filter five often produce regions which are very nearly the same (e.g. bounds differing by a few pixels). Any regions which whose centres are within a few pixels of each other are grouped as equivalent. Out of each equivalency group, the region whose bounding box is the smallest is selected, based on the assumption that the smallest box will be the tightest fit for the foreground area.

After filtering, typically only regions containing characters remain, sometimes along with a few spurious regions. An example is shown in figures 1 (before filtering) and 2 (after filtering). Note that I do not believe it is possible to distinguish between digits and characters without some form of machine learning.



Figure 1. Bounding boxes of MSERs detected in training image tr15.jpg.

1. House Number Extraction

Since it was not possible to filter out all nondigit regions, further processing is required to isolate the regions constituting the house number. This step begins by identifying plausible text paths, then selecting the path most likely to be the house number.



Figure 2. Bounding boxes of filtered regions from training image tr15.jpg.

The text path identification is inspired by [1]. For each region bounding box *B*0, the formation of a text path *P* beginning at *B0* and consisting of regions bounding boxes *B0*, *B1*, *B2*, etc. is attempted. A region with bounding box *Bx* is added to *P* if all of the following are true:

1. The angle from the centre of *B0* to the centre of *Bx* is in the range [-20°, 20°].
2. The distance between the centre of *Bx* and the centre of *Bx-1* in the X-axis is less than three times the maximum width of all boxes currently in *P*. (As done in [1].)
3. The absolute difference in height between *B*x and Bx-1 is less than or equal to 25% the height of *Bx*.

Condition 1 is derived from the observation that house numbers are roughly horizontal in the images. Condition 2 is derived from the observation that digits in the same number are spaced near to each other in the X-axis. Condition 3 is derived from the assumption that digits in the same number are roughly the same height, and that perspective warping present in the images is minimal.  
Regions are considered in the order of their X position, and the path is terminated if either of conditions 2 or 3 are violated. In this manner, plausible text paths are formed, extending to the right as far as possible.

Once text paths are formulated, the one containing the house number must be selected. Under the assumption that the house numbers contain the tallest characters, any paths with mean box height less than 75% of the largest mean box height are discarded. Of the remaining paths, the one containing the most regions is chosen.

1. Digit Recognition

The final step in the pipeline is classification of the digits constituting the house number. A linear SVM is used, as SVMs are known to be effective for digit recognition [1, 3, 4]. Neural networks, particularly convolutional neural networks, could also be effective, however no neural network libraries are available in the provided virtual machine environment. TODO

The SVM is trained on the provided labelled digit images, using OpenCV’s automatic training to optimise the SVM parameters via cross validation. Evaluation of the model was performed on the training data, since no other data was provided, with cross validation, and 100% accuracy is consistently achieved.

References TODO

[1] A Method for Text Localization and Recognition in Real-World Images

[2] Convolutional feature learning and Hybrid CNN-HMM for scene number recognition

[3] Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning

[4] Reading Digits in Natural Images with Unsupervised Feature Learning