TRAFFIC SIGN RECOGNITION SYSTEM

Blue Sun Software Australia

TABLE OF CONTENTS

INTRODUCTION

Purpose

The Traffic Sign Recognition system (from here referred to as ‘the library’) is a set of algorithms utilising AI and Computer Vision techniques to identify and categorize road signs.

The software is capable of recognizing and categorizing many types of road signs through various techniques. Signs are first identified within a frame, then their shape analyzed for what type of sign they may be, then scanned for images, text or otherwise from a bank of templates.

One all applicable algorithms have processed the sign, a set of confidence probabilities are assigned to each potentially detected sign, resulting in selection of what the application collectively thinks the sign is.

Approach

The approach taken utilizes a knowledge based AI technique of case based recognition, mimicking how humans interact with road signs.

The library looks for signs of different shapes, and then on seeing a shape will alter its approach with this new information accordingly. For example, if a square sign is seen then this is treated as a notification sign and then text is looked for that may disclose the notification. If a diamond is seen then this is a warning sign, the library will look within its memory for shapes that match the warning sign, and then be able to correlate what the sign is saying to be warned of. For circle signs these are speed regulation (in Australia) and these are immediately sent for processing by a modified OCR engine that only looks for numbers.

Technology

The library is a bundled iOS application designed to run on iPhone or iPad. The underlying algorithms however are mostly in C++ meaning that port to Android is possible, also as the application leverages OpenCV library the techniques used can be ported across to Python to be used in Raspberry Pi or similar microcomputer. The remainder of this document will focus on description of the existing application in its bundling with iOS, any items that will not carry across to other approaches are pointed out.

IMPLEMENTATION

Implementation Overview

The below diagram shows the basic overall process the library takes to identify and categorize road signs.

There are several steps for image processing – initially the frame is scanned for any interesting shapes, before these shapes themselves are passed to separate engines for dedicated processing.

Each utilized engine returns a confidence score which is then matched against database information/metadata for a final confidence score to be returned to the user.

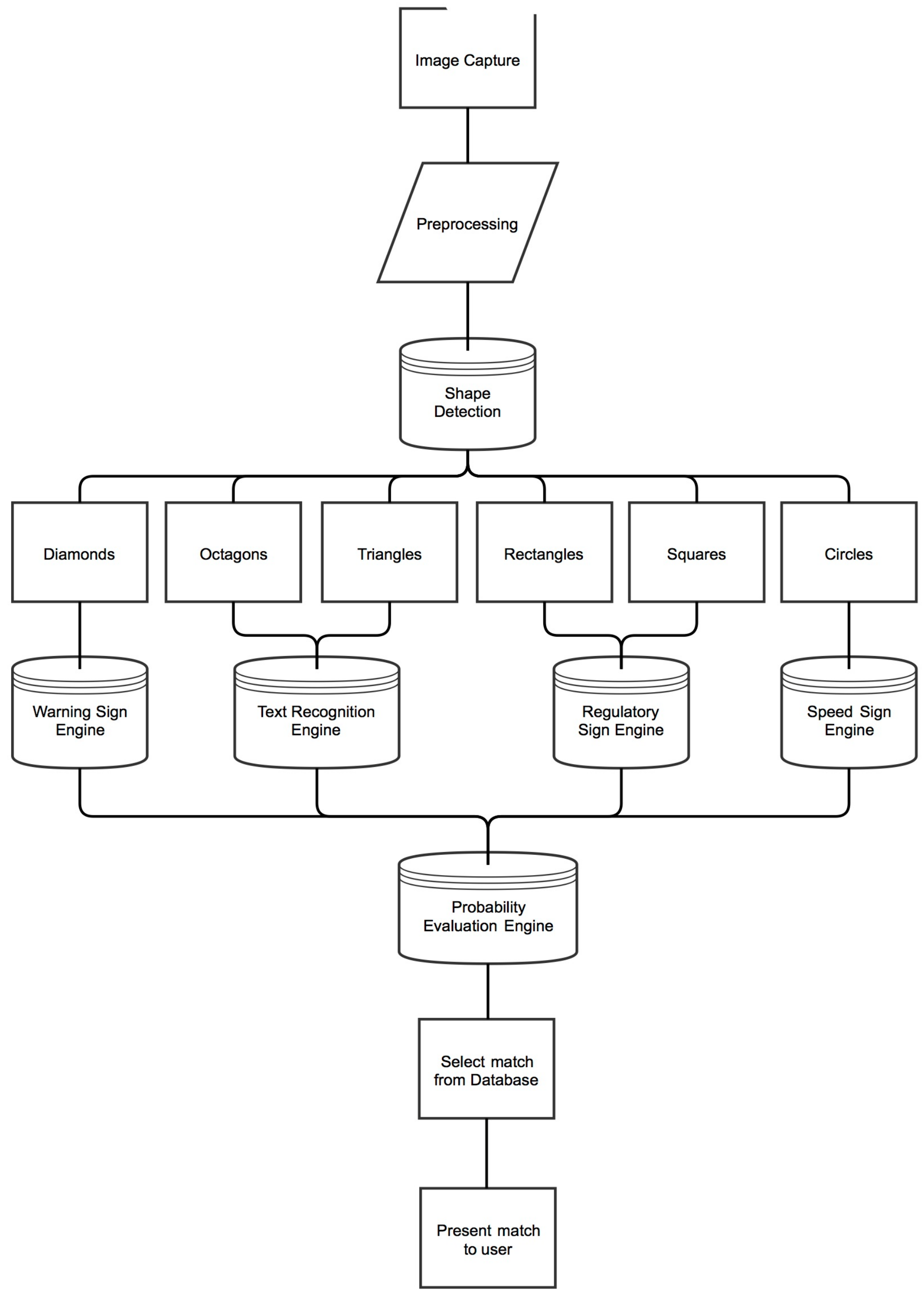


Figure 1: End to end library process

Image Capture

The library can use either the device stills camera or video camera. Video stills camera is recommended where battery life of the device is a concern:

e.g. with asset cataloguing the device may only take a single frame of data when the operator is in front of the sign. When device battery life is not a concern or constant scanning is required, the library can be configured to stream images from the device camera, although this is heavily dependent on the processing power available (benchmarking on iPad Mini 2 showed frame rates of ~10fps).

Image Preprocessing

The library takes the camera from a (or video camera frame) and applies preprocessing to the image in order to better performance for the computer vision algorithms.

Shape Detection

Primary shape detection is designed to be fast and robust: the library here is looking for 'interesting' shapes within the image as a whole. Interesting shapes are defined (in Australia) as:

- Diamonds (Warning signs e.g. Kangaroos)
- Squares/Rectangles (Notification signs e.g. 'Camden 5 km ahead')
- Octagon (Stop signs or railway crossing)
- Circles (Speed regulatory signs)
- Triangles (Give Way and some rarely used warning signs)

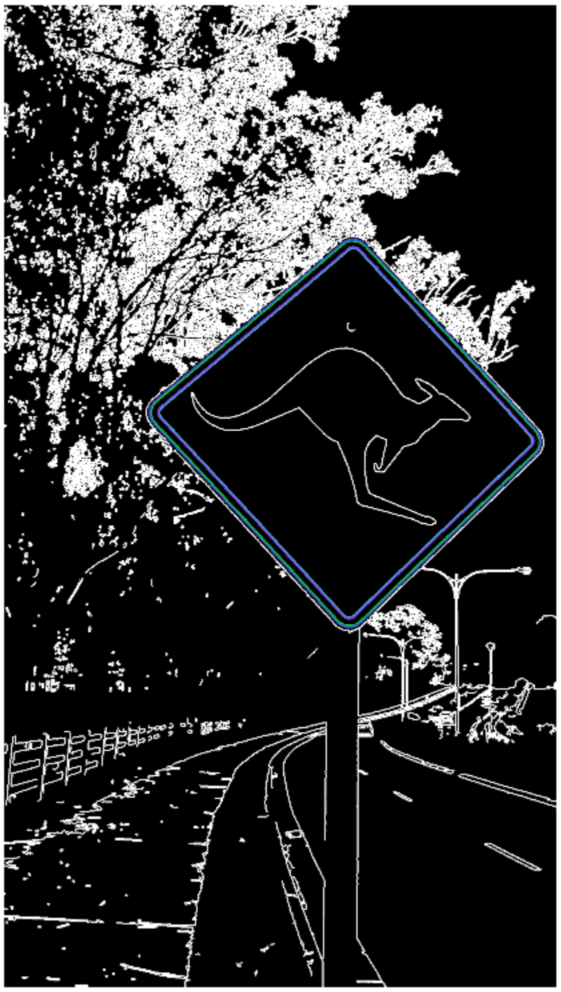


Figure 2: raw-out image processing library showing shape detection of a 'Kangaroo Warning' sign. Outer blue diamond is then sent to sub-engine for further processing.

Warning Sign Engine (Diamonds)

The warning sign engine takes the diamond and its contents, then sends this for specific processing to detect shapes from a library of preprocessed templates.

Template images are loaded when the application starts, and consist of a black and white basic representation of the shape that needs to be recognized. For example, with the kangaroo above the template is a simple black and white representation of the kangaroo. The templates are loaded on start and processed using computer vision to detect the shape itself using scale invariant methods. The resulting metadata is stored in application cache for rapid access during runtime.

During runtime if one of these shapes is potentially detected, the shape data is then compared with that loaded in template cache and compared, returning a confidence score. The top confidence scores are then returned to the probability engine at process end.

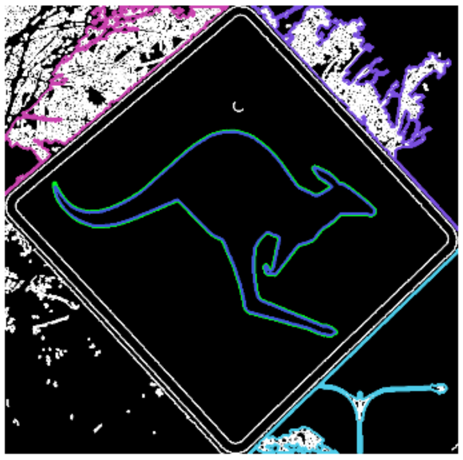


Figure 3: sample output sent to the Warning Sign algorithm, showing detecting of the inner kangaroo symbol, which is then sent for template matching.

Text Recognition Engine (Octagon, Triangle)

The text recognition engine processes all signs that may contain text.

This algorithm takes the contents and then processes, looking for internal features that satisfy ratios that would imply they are letters or text. The suspected text is then processed for text and characters come back with a probability score.

The highest probability scores are picked, and then the words that are detected are sent to the iOS auto-completion and spellcheck module (iOS feature only) in order to help deal with any problems that may have occurred. The auto-completion module is pre-loaded with a list of 'acceptable' words that may exist in the problem realm: e.g. list of words includes (not limited to) [Stop, Give Way, Warning, Crossing'] etc.

Regulatory Sign Engine

The Regulatory Sign engine works as a combination of the Warning (Shape Detection) and Text detection engines. These signs are sent for dual-processing and the probability of results returned.

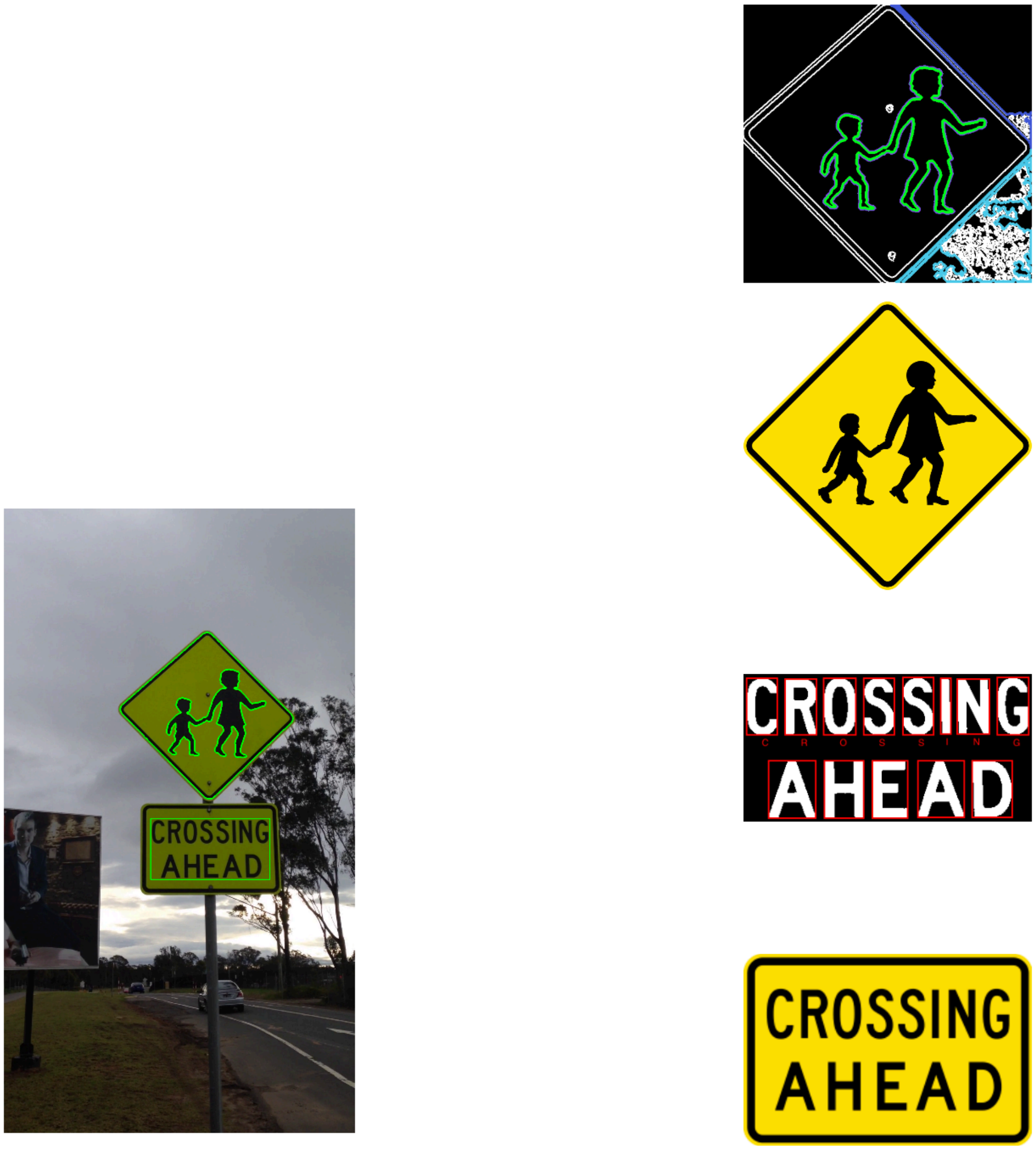
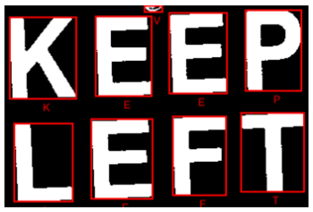


Figure 4: Multiple sign detection. Children Crossing warning sign detected via shape detection, 'Crossing Ahead' detected by regulatory sign text recognition algorithm.



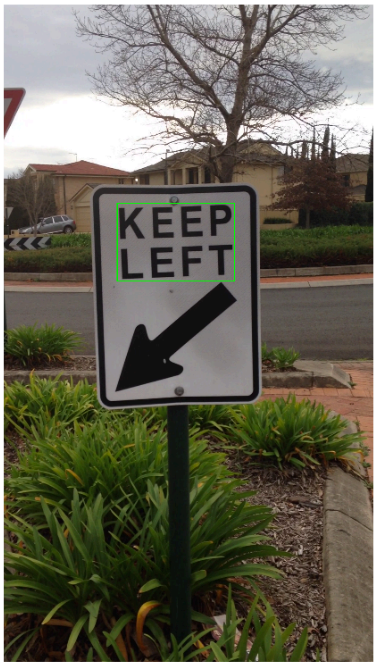
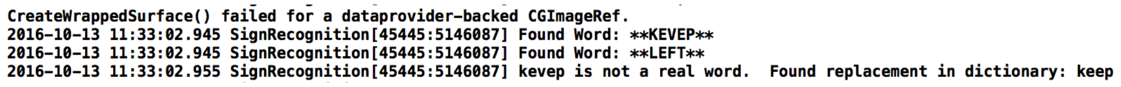


Figure 5: Detection of 'Keep Left'. Here the bolt-hole is within the text, causing the algorithm a problem as the bolt-hole is detected as a 'v' character (with low probability)

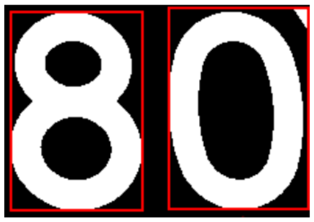


```
CreateWrappedSurface() failed for a dataprovider-backed CGImageRef.
2016-10-13 11:33:02.945 SignRecognition[45445:5146087] Found Word: **KEVEP**
2016-10-13 11:33:02.945 SignRecognition[45445:5146087] Found Word: **LEFT**
2016-10-13 11:33:02.955 SignRecognition[45445:5146087] kevep is not a real word.  Found replacement in dictionary: keep
```

Figure 6: 'Keep Left' is read as 'Kevep Left'. However, text processing shows no match in the library, finds closest permittable replacement as 'Keep'.

Speed Sign Engine

Any circles found in initial scan are sent to the speed sign recognizer. The speed sign recognizer looks for the thick red circle within the sign common with Australian speed warning signs, and then takes the internal data, reprocessing the image data and sending the numbers across to a special text recognition feature that only recognizes numbers.



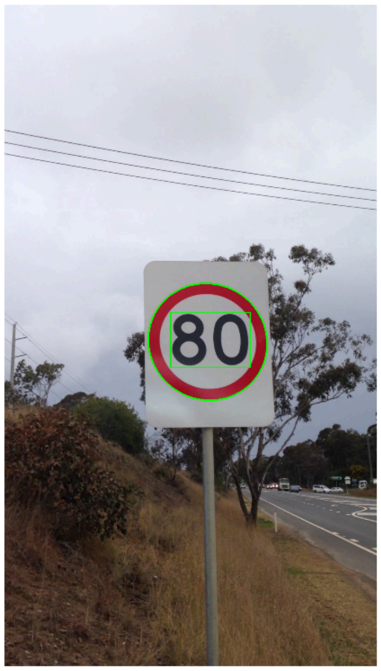


Figure 7: Detection of 80kmh sign

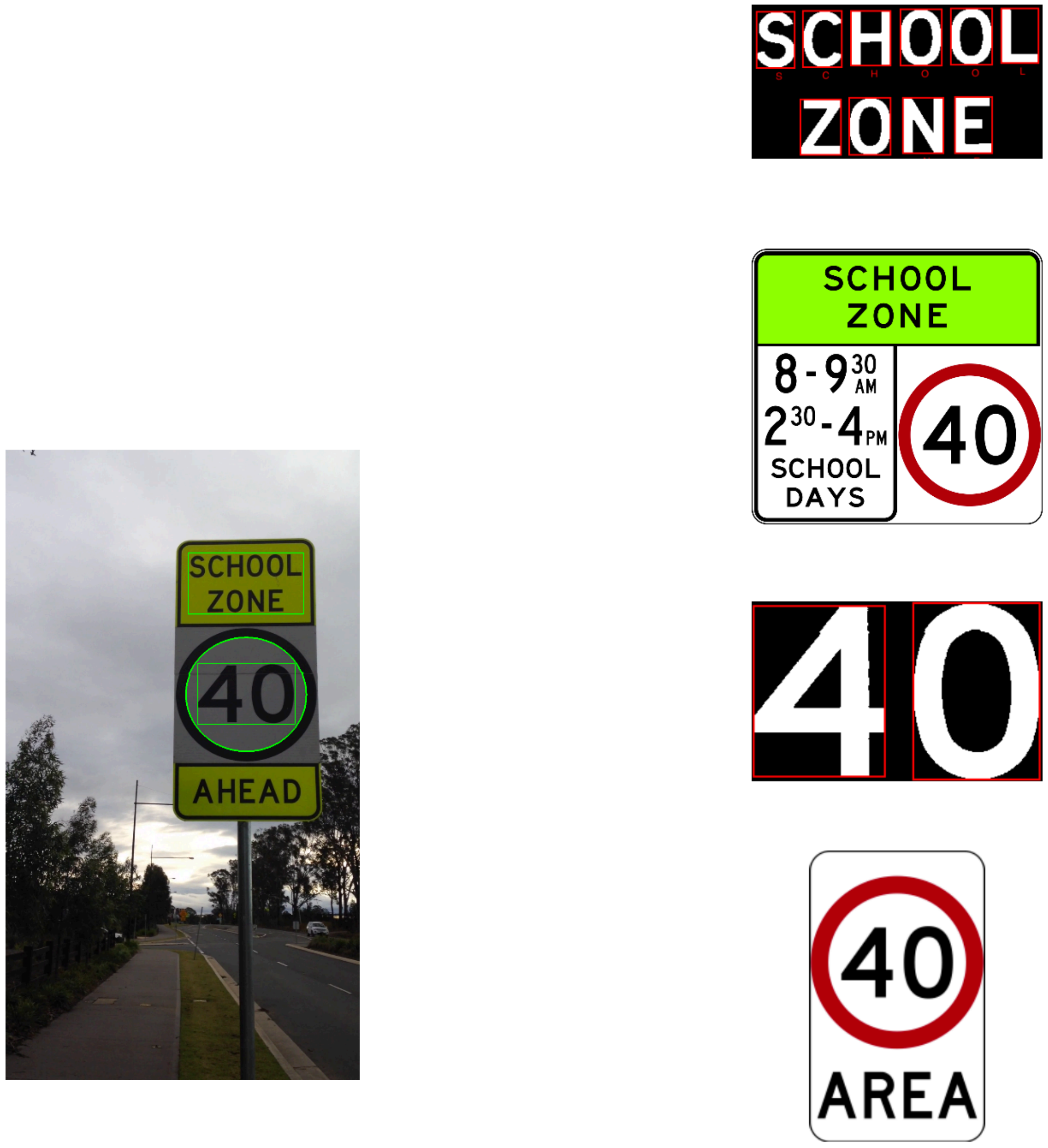


Figure 8: combination detection of 'School Zone' and Speed Limit

The speed sign detection engine assigns a probability to be assigned to the final step.

Probability Evaluation

Once all engines have run their course, all probability statistics are fed into a final probability engine of which infers which the most likely match (if any) is in front.

If no match can be found then the algorithm will take the next frame if in video mode, or prompt the user for a better photo if in camera/stills mode.