Test Bed for Number Plate Recognition Applications

D.G. Bailey*, D. Irecki*, B.K. Lim+ and L. Yang+
*Institute of Information Sciences and Technology, Massey University,
Private Bag 11 222, Palmerston North 5301, New Zealand
+School of Electrical and Electronic Engineering, Singapore Polytechnic,
500 Dover Road, Singapore 139651
D.G.Bailey@massey.ac.nz, BKLim@sp.edu.sg

Abstract

A flexible software based platform for number plate recognitions applications is described. It breaks the processing into several explicit modules, with the implementation for each module provided by a software plugin using a DLL interface. The modular structure greatly facilitates the comparison between different algorithms for a particular step, and allows code to be reused between applications.

1. Introduction

There are many applications for number plate recognition. Examples are:

- Border crossing control
- Identification of stole vehicles
- Automated parking attendant
- Petrol station forecourt surveillance
- Red light camera
- Speed enforcement
- Security
- Customer identification enabling personalised service

For many of these applications, much of the basic processing remains the same. However there may be environmental differences that necessitate changing some of the steps to make the algorithms more robust. Different countries and states also have different standards and use different fonts within number plates. These differences require some steps to be changed for different locales.

For these reasons, it was decided to develop a flexible test bed that could easily be adapted and allow different algorithms to be tested. Two initial applications are currently being considered. The first is petrol station forecourt surveillance, and the second is an automated parking lot attendant.

2. Software Architecture

The tasks common to a range of licence plate recognition applications were identified. To maximise the flexibility of the recognition system, a modular structure was chosen, with one module corresponding to each task or subtask. Seven processing modules have been identified as being common to all number plate recognition applications. While not every application will require every task, they provide a suitable partitioning of the overall application. The modules define the processing sequence as illustrated in figure 1. To allow modules to be interchanged, each module has specified inputs and outputs for the passing of data. Any additional module-specific parameters are provided wither through a configuration file, or through a user-dialog when the module is initialised. The detailed function of each of the modules will be described in turn.

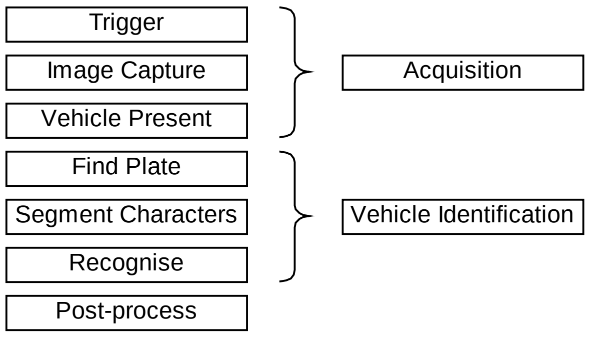

Figure 1: Software modules

The first group of modules deal with vehicle and image acquisition. It obtains the images, and ensures that the image contains a vehicle.

Trigger: The trigger indicates when an image should be captured. In many applications, the capture and processing of images would be activated by a hardware trigger and this module provides the software interface to that hardware (for example an inductive loop sensor within the vehicle lane). In other applications, it may be inconvenient or impractical to have a hardware trigger. In this case, the corresponding trigger module would always return true indicating that images should be captured and processed for a software trigger.

Image capture: Captures an image from a camera, or loads a file into the system either from disk or from over a network. This module interfaces with the specific image capture hardware or other image source.

Vehicle present: Determines if the current image contains a vehicle. If a hardware trigger system is used, this module just returns true, indicating that the hardware system detected the presence of a vehicle. However, in systems where a hardware trigger is impractical, this module effectively provides a software trigger. In such cases, an effective approach is to compare the current image with a background image, and detect if there are significant changes. If a vehicle is detected, processing then proceeds to the vehicle identification stage, otherwise the vehicle acquisition stage is repeated.

The next set of modules process the image obtained from the acquisition stage, and identifies the vehicle that has been detected.

Find plate: Locates the region in the image that contains the number plate. The return from this module is a set of candidate licence plate regions within the input image. In some applications, this is the first step of the main processing. In other applications, this step may be used to control the pan and zoom of the camera to obtain a higher resolution image of the licence plate.

A number of different techniques can be used for localising the registration plate, including colour detection [1], signature analysis [2] and edge detection [3].

Segment characters: Segments the individual characters in the number plate into separate binary images. The characters are taken from within the region provided by the find plate step. Segmentation is usually governed by size and relative location, with some form of dynamic thresholding required to minimise the effects of shadows [4].

Recognise: Performs OCR on each character. Techniques have been used for this step include template matching [4], feature matching [5] and neural network classifiers [6]. Rule based methods may also be used to help resolve ambiguities, for example between the digit ‘0’ and the letter ‘O’.

The find plate, segmentation and recognition steps are split into different modules because these steps are often distinct in many of the algorithms that are used. Separating the processing into distinct modules allows different combinations of the techniques to be evaluated.

Post process: The final module provides application specific processing of the resulting number plate string. Depending on the application, this may be to save the image (in security applications), look the plate up in a database (to identify stolen vehicles, or regular customers), to start timing or produce an invoice or account (parking applications), or to issue a traffic infringement notice (traffic monitoring).

Rather than have all of the modules compiled directly in with the main program, they have been implemented as software plugins through dynamic link libraries (DLLs). This allows the modules to be developed independently, and be switched easily when refining the algorithms. It also facilitates code reuse by allowing techniques that have worked in one licence plate recognition application to easily be tried in another application.

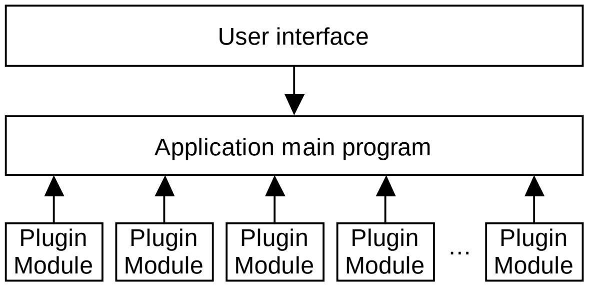

Figure 2: Plugin structure

The plugin structure is illustrated in figure 2. When the application main program is run, it automatically loads all of the DLLs present in the current directory, making the routines in them available to the main program for processing.

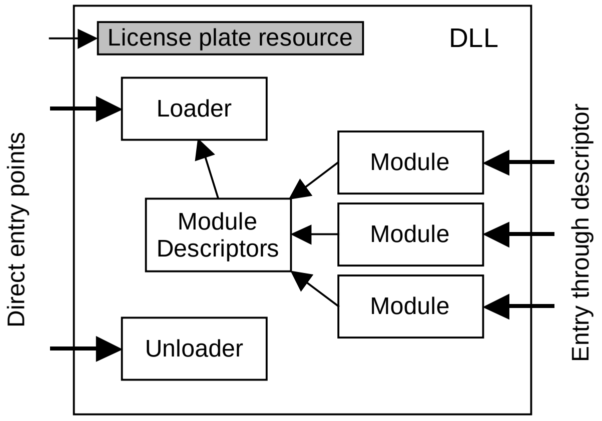

Figure 3: DLL structure

The structure of the DLLs is as shown in figure 3. Each licence plate DLL contains a resource that indicates to the main program that it contains licence plate modules. The presence of this resource is checked prior to loading the DLL. Since each DLL may hold several modules, it also contains a loader and unloader routine.

The loader provides a descriptor for each module indicating its name, function, and calling address. This information allows the main program to select and call the procedure in the DLL. The unloader frees any resources that may have been allocated by the routines in the DLL, and prepares the DLL for being removed from memory.

Multiple implementations of the same module are placed within menus to allow the modules to be switched quickly and easily. This provides a simple mechanism for investigating different algorithms and techniques. Modules may also be unloaded, recompiled and reloaded without exiting the main program. This speeds development of the software because it can be tested and modified in place, avoiding the reconfiguration each time the program is run.

The modular software architecture also makes it easier to save a range of images and transparently test the operation by simulating image capture through loading the images from file. Changing from online to offline mode and back again can be done through a menu selection of the appropriate image capture module. Working in off-line mode allows different algorithms to be easily compared on the same set of images, facilitating a direct comparison of the effectiveness of each algorithm.

The main program also provides the user interface, which displays the status of the system and all of the intermediate images required during development to evaluate the effectiveness of the modules in a particular application or scenario.

3. System Testing

To demonstrate the system and test the initial algorithms, a test site is currently being established on the campus of Singapore Polytechnic. This test site is situated on the entrance of an outdoor car park. Initial tests will be performed during daylight under favourable weather conditions, with hardware triggering to indicate the presence of a vehicle. Once this system is successfully demonstrated, the various restrictions will be lifted. The eventual goal is a fully automated all-weather parking attendant using software based vehicle detection.

Initial testing has identified incompatibilities between different compilers. Many of these problems are minimised by using a common module for allocation and deletion of all data structures. In the initial system, the modules are as follows:

Trigger: Hardware based, using an inductive sensor. For software based triggering, a dummy module that always returns true will replace this.

Image capture: Two modules here: the first captures an image from a camera, and the second loads an image from disk for system testing.

Vehicle present: For hardware triggering, this is a dummy module. For software triggering, this will compare the image with a background for vehicle detection. If no vehicle is detected, this would update the background image, to enable varying light conditions to be handled.

Find plate: The initial method is based on detecting clusters of strong edges with an aspect ratio of that of licence plates [4].

Segment characters: Within the region of interest, a morphological filter is used to automatically select a dynamic threshold. Noise is removed based on object size, shape, and alignment [4].

Recognise: Fuzzy template matching will be used to identify normalised characters [4].

Post process: During initial testing, the images and plate identification will be saved for evaluating system limitations. Later, post processing will control a barrier arm, allowing only authorised vehicles into the park.

4. Summary

A modular structure will facilitate the testing and developing of software for vehicle number plate reading applications. The breakdown of the system, and the plugin mechanisms are described. The flexibility and modular nature of the system simplifies development and testing, and allows modules developed for one application to easily be reused in other applications.

5. Acknowledgements

The authors would like to acknowledge financial support from the Asia 2000 Foundation Higher Education Exchange Program, and from Technology New Zealand through a Technology for Industry Fellowship.

6. References

[1] B. H. Cho & S. H. Jung, “Non feature-based vehicle plate recognition system using neural network”, *Proceeding of ITC-CSCC 98 International Conference*, Korea, Vol. 2, 1065-1068, July 1998.

[2] J. Barroso, J. Bulas-Cruz & E. L. Dagless, “Real Time Number Plate Reading”, *4th IFAC Workshop on Algorithms and Architectures for Real-time Control*, , Portugal, April 1997.

[3] J. R. Parker & P. Federl, “An approach to licence plate recognition”, *Computer Science Technical reports,* University of Calgary, Alberta Canada, Vol. 591-11, October 1996.

[4] D. Irecki & D. G. Bailey, “Vehicle registration plate localization and recognition”, *Proceedings of the Electronics New Zealand Conference, ENZCon’01*, New Plymouth, New Zealand, September 2001.

[5] J. Barroso, A. Rafael, E. L. Dagless & J. Bulas-Cruz, “Number plate reading using computer vision”, *IEEE Intern. Symposium on Industrial Electronics, ISIE’97*, 1997.

[6] R. van Heerden and C. Nieuwoudt, “Automatic number plate segmentation and recognition”, *Department of Electrical and Electronic Engineering Reports,* University of Pretoria, 1998.