Traffic Signal Detection and Tracking

Mahmoud Abdallah
Daniel Eiland
Spring 2011

Project Overview

We have developed the following 3-stage process capable of identifying and tracking individual signals by utilizing heuristics gathered across multiple frames

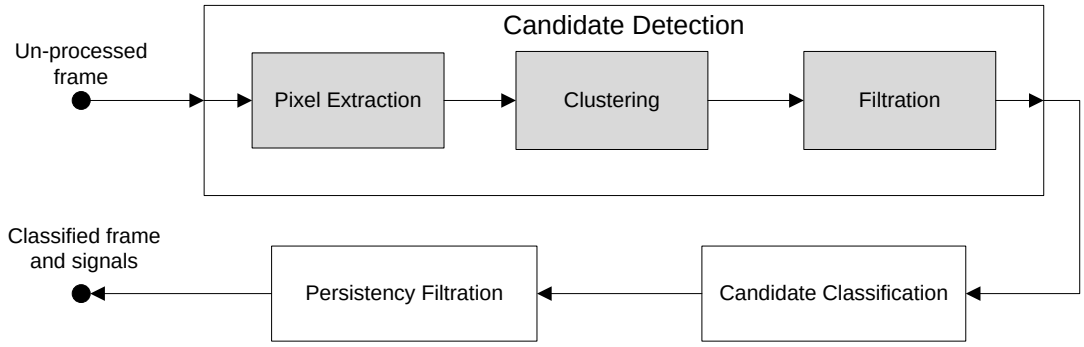

Figure 1 – High-Level Frame Processing

Candidate Detection

The first processing stage is responsible for identifying signals within the current frame and is broken into three phases. The first phase is the extraction of pixels that are may be part of a signal. Currently, pixels are extracted by comparing their Red and Green values against a high-threshold. The extracted pixels are then clustered into groups based on their connectivity[3]. Finally, the clusters are filtered based on their size and circular shape. The shape is calculated using the following compactness measurement:

$$C = \frac{Perimeter^2}{Area} - 4\pi$$

Where a lower C corresponds to a more circular shape.

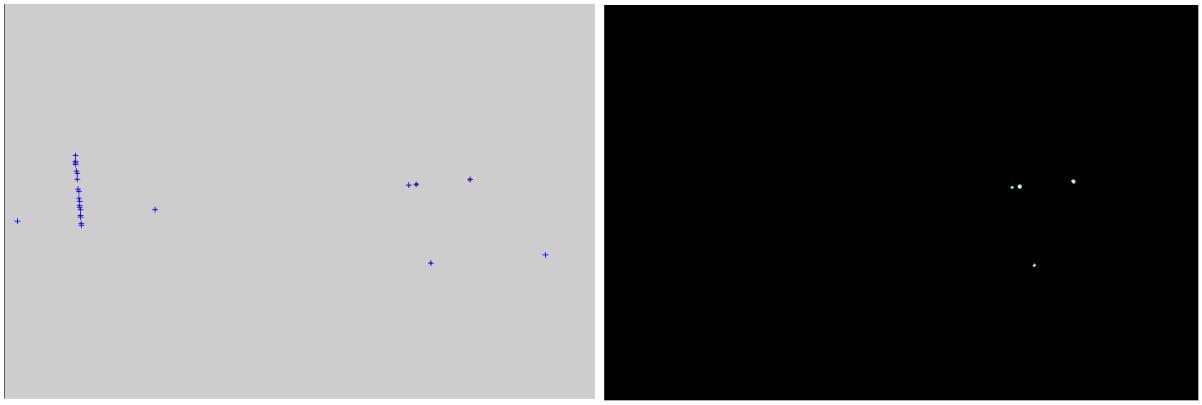

Figure 2 – Candidate Detection Results (Left to Right): Input Frame, Pixel Extraction, Clustering, Filtration

Candidate Classification

Once the set of candidates have been identified, their relationship to previously detected signals is determined. As frames are processed, signal groups are created that associate candidates using their centroid. As new candidates are detected, they are placed into the group with the closest proximity. If the candidate cannot be matched to an existing group, it is placed into a new group. This allows a given signal to be tracked across multiple frames.

When a candidate is placed into a group, its color is also detected (RED, YELLOW, or GREEN). The overall color of the group is then based on the candidate color that is detected the most.

Persistency Filtration

The final processing stage involves the filtration of candidates based on their rate of detection. Once a signal is first captured, it should appear across several frames based on the camera's rate-of-travel. However the first processing stage is sometimes over-zealous and removes candidates that are actual signals. This is overcome by adding simulated candidates to each signal group that was not matched in the previous stage. When a signal has a low detection rate, its group will contain mostly simulated candidates. This allows the persistency filter to classify the group as an anomaly and remove it before the final results are returned.

Distance estimation

In this phase, the distance approximation of camera from detected object is estimated by considering the geometry of two captured frames. Assuming the velocity of camera is known, the distance of camera is estimated according to the following approach:

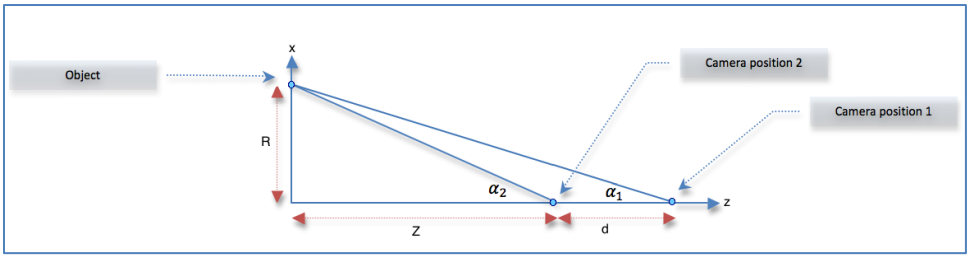

Figure 3 – Geometry of moving camera

From the figure above, we can define the following equations:

$$\frac{R}{Z} = tan\,\alpha_2\,,\ \frac{R}{Z+d} = tan\,\alpha_1 \qquad ...(1)$$

Thus,

$$R = Z\,tan\,\alpha_2 = (Z+d)\,tan\,\alpha_1$$

$$Z = \frac{d\,tan\,\alpha_1}{(tan\,\alpha_2 - tan\,\alpha_1)} \qquad ...(2)$$

Now, consider that the camera is fixed and the object (signal) has moved on the image plane.

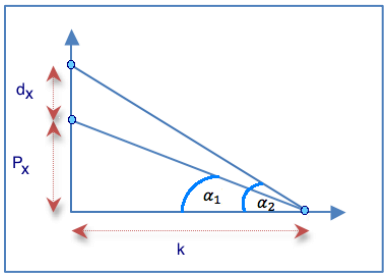

Figure 4 – Geometry of fixed camera

Define the following equations from figure (4):

$$\frac{P_x+d_x}{k} = tan\,\alpha_2\,,\ \frac{P_x}{k} = tan\,\alpha_1 \qquad ...(3)$$

Plugging equation (3) into equation (2) we get:

$$Z = \frac{d\,P_x}{d_x} \qquad ...(4)$$

Where P_x is the distance from object to the center of image in the first frame, d_x is the distance the object has moved in the second frame, and d is the distance the camera has moved. The distance d could be found from camera velocity.

Future Work

While the algorithm has shown a high detection rate in testing, several items have been identified as possible points of improvement.

The RGB color model is used in the identification of candidate pixels and color classification. Due to similar values for red and yellow lights (Red [233R, 85G, 88B]; Yellow [226R, 90G, 81B]), the classification of the signal's color is inaccurate. The use of a different color model (HSV or HSI) may provide better values for identifying the different signal types.

Because the primary metric used in signal identification is its (circular) shape, turn signals (→) are not classified. If a higher resolution camera were employed, the identification of the "arrow" shape would be possible using an additional metric (template matching or other feature-based metric).

While the current filtration metrics provide a high-detection rate, there are still cases of false-signal detection. Additional filters – such as the intensity of surrounding pixels – could be employed to remove candidates that are similar to an actual signal (such as tail lights or street lights).

References

1. Park, J. and Chang-sung, J., "Real-time Signal Light Detection"; International Journal for Signal Processing, Image Processing and Pattern Recognition; June 2009; Vol. 2, No. 2; http://www.sersc.org/journals/IJSIP/vol2_no2/1.pdf
2. Chung, Y., Wang, J. and Sei-Wang, C., "A Vision-Based Traffic Light Detection System at Intersections"; Journal of Taiwan Normal University: Mathematics, Science & Technology; 2002; Vol. 47; http://wjm.tyai.tyc.edu.tw/~jmwang/paper/product/mst471-4.pdf
3. Chang, F., Chen, C., and Lu, C., "A linear-time component labeling algorithm using contour tracing technique"; Journal of Computer Vision and Image Understanding; Feb 2004; Vol. 93, No. 2

Code Overview

Development Environment

- Language – C++
- OS – Window XP
- IDE/Compiler – Visual Studio 2010

Camera Type

Panasonic PV-GS250 (http://service.us.panasonic.com/OPERMANPDF/PVGS250.PDF)

Resolution: 720x480

Frames Per Second: 29

Primary Files

- lightmain.cpp – Main entry point into project. Reads video frames and passes to SignalClassifier for processing. Displays classification results.
- SignalClassifier.h / .cpp – Primary signal detection and tracking logic.
- SignalGroup.h / .cpp – Stores individual signals that have been detected across multiple frames.
- trajectoryUtility.h – Data structure used in tracking signals across frames.
- Common.h – #defines for modifying program behavior (See below)

External Dependencies

- OpenCV 2.2
    - Description: Computer Vision toolkit. Provides video I/O and image processing framework.
    - Location – http://opencv.willowgarage.com/wiki

    - Download
        - Windows Installer (Visual Studio 2010) – http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.2/OpenCV-2.2.0-win32-vs2010.exe/download
    - Files
        - Windows - opencv_calib3d220d.lib/.dll, opencv_contrib220d.lib/.dll, opencv_core220d.lib/.dll, opencv_features2d220d.lib/.dll, opencv_ffmpeg220d.lib/.dll, opencv_flann220d.lib/.dll, opencv_gpu220d.lib/.dll, opencv_highgui220d.lib/.dll, opencv_imgproc220d.lib/.dll, opencv_legacy220d.lib/.dll, opencv_ml220d.lib/.dll, opencv_objdetect220d.lib/.dll, opencv_ts220.lib/.dll, opencv_video220d.lib/.dll
- cvBlob
    - Description: Connected component analysis (i.e. Clustering)
    - Location - http://code.google.com/p/cvblob/
    - Source Download – http://cvblob.googlecode.com/files/cvblob-0.10.3-src.zip
        - This toolkit must be compiled for the specific platform before usage
    - Files
        - Windows – cvblob.lib/.dll
- CMake
    - Description: Cross platform build system. Required to create makefiles for cvBlob
    - Location – http://www.cmake.org
    - Download – http://www.cmake.org/files/v2.8/cmake-2.8.4-win32-x86.exe

Program Behavior – #define

The following #define statements found in Common.h can be modified to easily change the detection behavior.

Name	Valid Values	Behavior
OFFSET	0.0 < x < INFINITY	The maximum offset between
MIN_SIGNAL_SIZE	0 < x < MAX_SIGNAL_SIZE	Minimum detection size (in pixels)
MAX_SIGNAL_SIZE	MIN_SIGNAL_SIZE < x < INFINITY	Maximum detection size (in pixels)
VERTICAL_PORTION	0.0 < x <= 1.0	Determines the percentage of rows (from a frame) that will be analyzed for signals
BUFFER_SIZE	1 < x < INFINITY	Number of frames buffered during classification
DETECTION_RATE	0.0 < x <= 1.0	The (%) rate at which a candidate must be detected to be classified as a signal
GREEN_THRESHOLD	0 < x < 255	The minimum GREEN value of candidate pixel
RED_THRESHOLD	0 < x < 255	The minimum RED value of a candidate pixel
SMOOTHING_FACTOR	0.0 < x <= 1.0	The weight given to the raw motion value during motion calculation (0.0 → less; 1.0

		→ more)
CAMERAVELOCITY	0.0 < x < INFINITY	The speed of the camera used in distance calculations.