Traffic Sign Recognition for Intelligent Vehicle/Driver Assistance System Using Neural Network on OpenCV

Auranuch Lorsakul, Jackrit Suthakorn
Mahidol University

URAI 2007

Motivation

- Why use traffic sign recognition?
  - Make driving easier and safer.
  - Important for automated vehicles
- Requirements
  - Runs in real time
  - Accurately identify signs in varied environments
- Current TSR systems are too slow or not accurate enough.

Previous Work

- PROMETHEUS (1986, Daimler-Benz research)
  - Commercial
- Pacheco et al. (1994)
  - Used colored barcodes under signs to aid recognition
  - Not cheap, not extensible
- Aoyagi and Asura (1996)
  - Converted images to grayscale
  - Hurt robustness

Previous Work

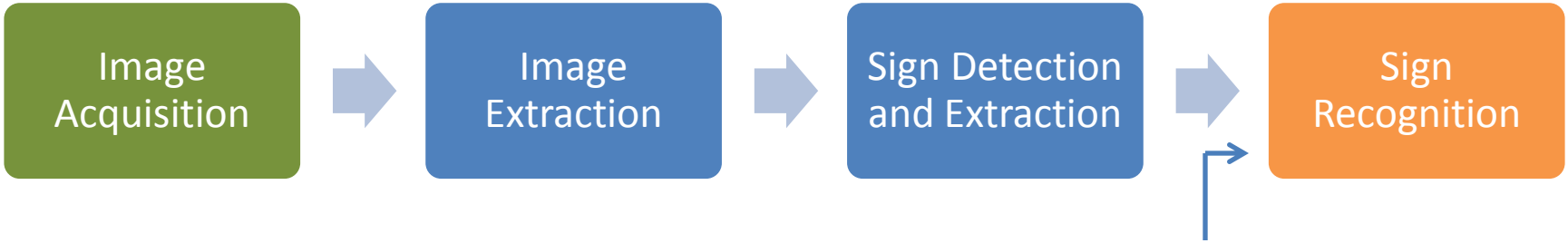
- TSR systems generally develop with two separate phases
  - Detect and extract traffic signs from a video feed, based on shapes and colors
  - Recognize signs using a neural network
- Common Problems
  - Environment: lighting, shadows, air quality, weather
  - Distortions: Motion blur, vibration, contrast changes

Proposal

- Implement a MLP that can recognize traffic signs in real time, to be used in an intelligent vehicle
  - Detection Phase: identify regions of the image that might be signs
  - Recognition Phase: MLP recognizes and classifies the image region as a particular sign
  - Use OpenCV to handle image processing

System Design

Image Sequence *Images* *Blobs*



Sign Detection and Extraction

- Takes in a single image, from the image extractor
- Creates a binary image used to detect contours
  - Converts RGB to greyscale
  - Gaussian smoothing to reduce noise
  - Canny Edge Detection to remove background data, highlight image structure


Photo by Tristan Savatier - www.loupiote.com

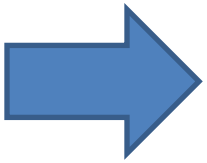

Photo by Tristan Savatier - www.loupiote.com

Sign Detection and Extraction

- Binary image is searched for contours
- Contours used to find a bounding ellipse for the sign, using OpenCV's cvFitEllipse function
- Bounding ellipse center and size used to extract a section of the original image
- Image chunk resized to 30x30 pixels, converted to greyscale

Sign Detection and Extraction

Original Image



“Blob”

Form Recognition

- How to input an image to a neural network?
  - 2700 input bits (30 x 30 x 3 bits/pixel)?
  - Average RGB values
  - Greyscale intensity histogram

$$vh_i = \frac{1}{30}\sum_{i=1}^{30}(b'_{i,j} > T)$$

$$hh_i = \frac{1}{30}\sum_{j=1}^{30}(b'_{i,j} > T)$$

$$T = \frac{1}{900}\sum_{i=1}^{30}\sum_{j=1}^{30} b'_{i,j}$$

Form Recognition

- MLP Architecture
  - 63 Inputs
  - C outputs (Exact number was unspecified)
  - How many nodes in the hidden layer?
    - Use cross-validation to find the optimal network
    - 42 was their optimal number of hidden nodes
  - Binary sigmoid activation function
  - Error measured with Least Squares and Kullback-Leibler Divergence

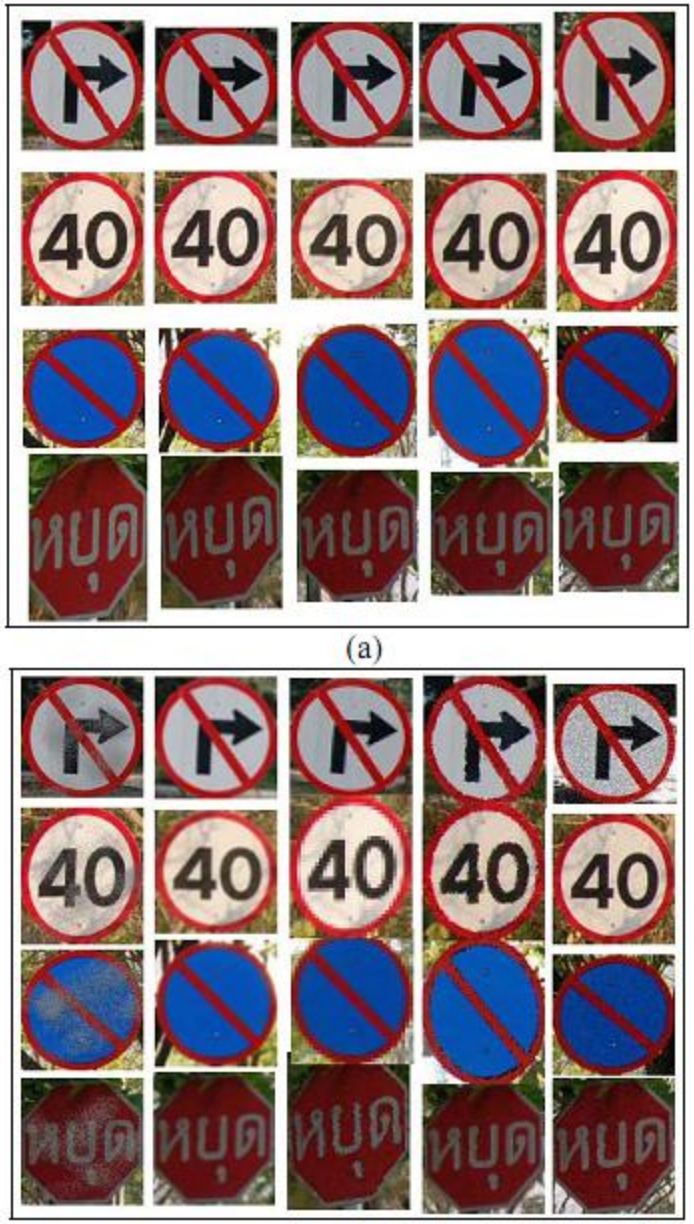
Error Measures

- Least Squares Criterion

$$\frac{1}{Q}\sum_{q=1}^{Q}\sum_{r=1}^{R}\left(y^{(r)}(q)-f^{(r)}\left(x(q),w(q)\right)\right)^2$$

- Kullback-Leibler Divergence

$$-\frac{1}{Q}\sum_{q=1}^{Q}\sum_{r=1}^{R}\log\left[1-\left|y^{(r)}(q)-f^{(r)}\left(x(q),w(q)\right)\right|\right]$$



(a)

(b)

Fig. 7. (a) Training data without problems, and (b) Training data with distortion problems

Data

- Training data had two sets
  - Clean data
  - Data that had been distorted in some way (blur, color shift)
- Validation / Testing set separate from training data

Fig. 8. Validating data

Results

- Test Set = 52 images

Least Squares Error	0.0865
K-L Divergence	0.3231
Average Processing Time	37.27 ms (>25 images/sec)

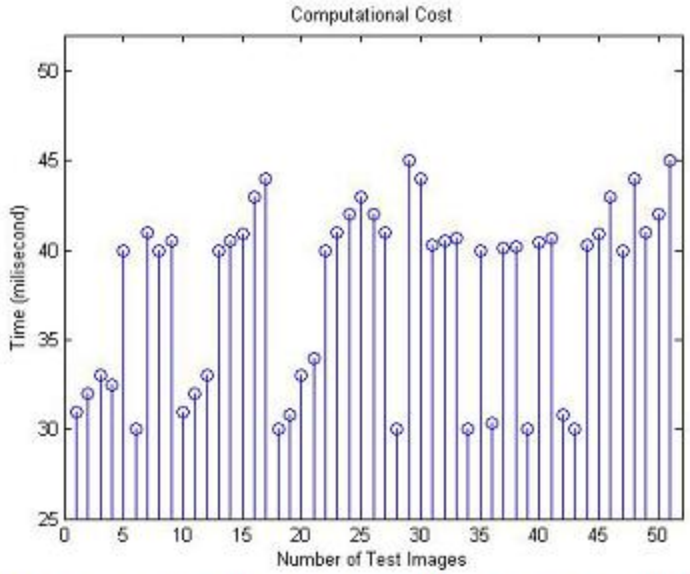


Fig 12. Computational Cost of the proposed method

Results

- Test Set = 52 images

Least Squares Error	0.0865
K-L Divergence	0.3231
Average Processing Time	37.27 ms (>25 images/sec)

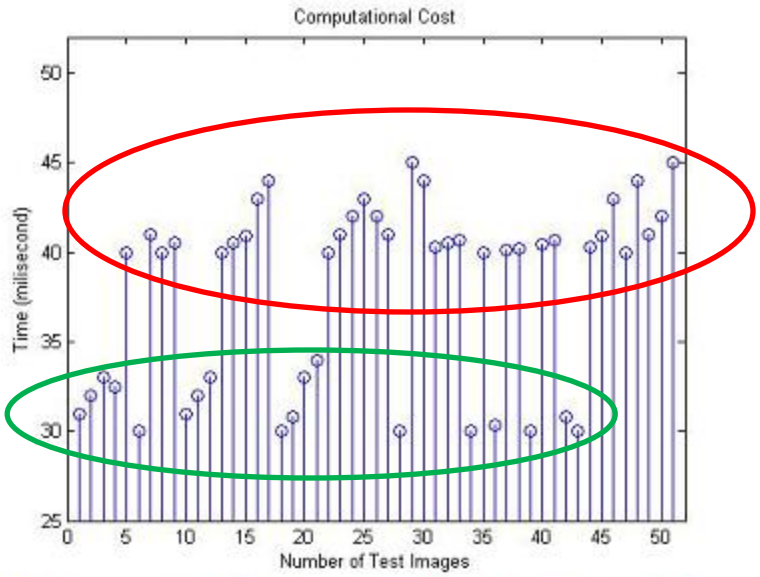


Fig 12. Computational Cost of the proposed method

Conclusions

- This method effectively reduces the computational cost of traffic sign recognition, allowing a real-time implementation.

Questions?