

PHÁT HIỆN VÀ NHẬN DẠNG MỘT SỐ BIỂN BÁO GIAO THÔNG ĐƯỜNG BỘ NGUY HIỂM TẠI VIỆT NAM

Lê Chân Thiện Tâm, Phạm Hồng Thái

Khoa Công Nghệ Thông Tin,

Đại học Lạc Hồng

{thientamlhu,hongthaipro}@gmail.com

Trần Tiến Đức

Khoa Công Nghệ Thông Tin,

Đại học Sư Phạm Kỹ Thuật TP. HCM

ductt@fit.hcmute.edu.vn

Tóm tắt:

Trong bài báo này, chúng tôi trình bày phương pháp phát hiện và nhận dạng biển báo giao thông. Nhận dạng biển báo giao thông là vấn đề quan trọng vì nó hỗ trợ người tài xế ý thức và chủ động hơn trong việc xử lý các tình huống nguy hiểm tiềm ẩn khi điều khiển phương tiện lưu thông. Biển báo giao thông được phát hiện bằng phương pháp rút đặc trưng Haar kết hợp với bộ huấn luyện tăng tốc Adaboost. Sau đó rút đặc trưng bằng phương pháp Principle Components Analysis và được nhận dạng bằng thuật toán Support Vector Machine. Hơn 1800 biển báo giao thông của 10 loại biển được chụp trong những điều kiện khác nhau như trong môi trường ánh nắng gắt, thời tiết mát, thời tiết âm u nhiều mây, biển báo bị che khuất bởi bóng râm, bị hư hại một phần, trong quang cảnh đường nông thôn, đường thành phố đông xe. Trên 500 ảnh thử được dùng, kết quả nhận dạng có độ chính xác trung bình là 92,97% cho phép kết luận hướng nghiên cứu này là thích hợp.

I. GIỚI THIỆU

Giao thông Việt Nam luôn là một trong những vấn đề nóng của xã hội. Theo số liệu mới nhất từ Ủy ban An toàn giao thông quốc gia, chỉ trong 8 tháng đầu năm 2012, Việt Nam có hơn 7000 vụ tai nạn giao thông [10], dân số mất đi hơn 6000 người. Từ tình hình thực tế giao thông tại Việt Nam, nguyên nhân phần lớn các vụ tai nạn giao thông đường bộ là do tài xế lái xe không làm chủ tốc độ, không quan sát hoặc không kịp nhận ra các loại biển báo và tín hiệu giao thông. Đây là những nguy hiểm đã được cảnh báo trước nhưng tai nạn vẫn thường xuyên xảy ra. Chúng tôi mong muốn góp phần hạn chế những tai nạn giao thông và giảm thiểu hóa những hậu quả sau tai nạn, hỗ trợ người tài xế ý thức và chủ động hơn trong việc xử lý các tình huống nguy hiểm tiềm ẩn khi điều khiển phương tiện lưu thông.

II. ĐẶT VẤN ĐỀ

Cho đến nay vấn đề này được nhiều nghiên cứu trên thế giới quan tâm [1],[2],[4]. Nhưng những biển báo giao thông được nghiên cứu không phải dùng cho giao thông đường bộ tại Việt Nam. Trong khi đó tình hình nghiên cứu biển báo giao thông tại Việt Nam vẫn còn nhiều hạn chế [3], [5] và chưa đầy đủ.

III. NỘI DUNG NGHIÊN CỨU

1. Trích đặc trưng Haar-like

Một trong những kĩ thuật quan trọng được sử dụng phổ biến trong việc nhận dạng đối tượng là kĩ thuật dựa trên đặc trưng Haar-like được công bố bởi Viola và Jones [8]. Đặc trưng Haar-like gồm 4 đặc trưng cơ bản để xác định một đối tượng trong ảnh. Những khối đặc trưng này thể hiện sự liên hệ tương quan giữa các bộ phận trong ảnh mà bản thân từng giá trị pixel không thể diễn đạt được.



Hình 1. Bốn đặc trưng Haar-like cơ bản.

Để tính toán giá trị đặc trưng Haar-like cần phải tính toán tổng các vùng pixel trên ảnh. Do đó để có thể đáp ứng yêu cầu về xử lý thời gian thực Viola và Jones đã trình bày khái niệm “Integral Image” để giải quyết bài toán. “Integral Image” tại vị trí x, y được tính theo công thức (1).



$$P(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (1)$$

Hình 2. Cách tính Integral Image của ảnh

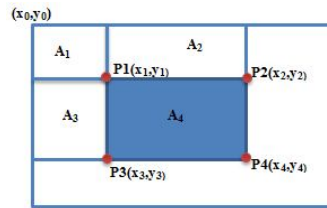
Kết quả có được sau khi tính Integral Image, việc tính tổng giá trị pixel trong vùng cần tính thực hiện như sau:

Gọi vùng cần tính tổng các giá trị pixel là vùng “ A_4 ”.

$$P_1(x_1, y_1) = A_1; \quad P_2(x_2, y_2) = A_1 + A_2;$$

$$P_3(x_3, y_3) = A_1 + A_3; \quad P_4(x_4, y_4) = A_1 + A_2 + A_3 + A_4;$$

$$A_4 = P_4 + P_1 - P_2 - P_3; \quad (2)$$



Hình 3. Cách tính tổng giá trị pixel vùng cần tính

2. Adaboost

AdaBoost (*Adaptive Boost*) là một bộ phân loại phi tuyến mạnh cải tiến từ thuật toán Boosting, giúp đẩy nhanh việc tạo ra bộ phân loại mạnh (*strong classifier*) bằng cách chọn các đặc trưng tốt Haar-Like trong bộ phân loại yếu (*weak classifier*) và kết hợp chúng lại tuyến tính

để hình thành một bộ phân loại mạnh (*strong classifier*) bằng cách sử dụng các trọng số (*Weight*) để đánh dấu các mẫu khó nhận dạng.

Cho trước một vài thuật toán học yếu, người ta áp dụng kỹ thuật tăng cường để tạo ra thuật toán học mạnh hơn. Adaboost là thuật toán cho phép ta có thể làm được điều trên. Gồm có 2 giai đoạn chính như sau [9]:

Giai đoạn 1: Tạo ra các giả định yếu (*weak hypotheses*) từ các thuật toán học yếu.

Giai đoạn 2: Tạo ra các thuật toán học mạnh từ các giả định yếu.

Với một tập dữ liệu huấn luyện, một giả định yếu được khởi tạo như sau:

Đầu vào(Input): Cho một tập dữ liệu huấn luyện, N cặp (x_i, y_i) , x_i là các vector đặc trưng Haar, y_i là giá trị đầu ra mong muốn luôn luôn mang giá trị bằng +1 hoặc -1 (trong đó +1 là đối tượng (object), -1 không phải là đối tượng (background)), và số lượng vòng lặp là T .

Đầu ra(Output): Tồn tại một hàm $f_T(x)$ có thể được sử dụng để phân lớp các đặc trưng của vector x .

Nếu $f_T(x) < 0$ thì x được phân lớp là -1.

Nếu $f_T(x) > 0$ thì x được phân lớp là +1.

Khởi tạo(Initialization): Cho trọng số $W_i = \frac{1}{N}$

Lặp(Iterate): For $t = 1, \dots, T$ tính giả định (*hypothesis*) h_t , trọng số tốt (*goodness*) α_t , và đồng thời cập nhật lại các trọng số W_1, \dots, W_N theo các bước sau:

Bước 1: Chọn ngẫu nhiên một tập con S_t trong tập dữ liệu huấn luyện. Trong trường hợp này chính là trọng số W_i .

Bước 2: Tính toán giả định h_t bằng cách sử dụng bộ phân lớp yếu cho S_t .

Bước 3: Tính toán sai số của trọng số huấn luyện ε_t của h_t :
$$\varepsilon_t = \sum_{y_i \neq h_t(x_i)} P_i$$

Bước 3.1: Nếu $\varepsilon_t \geq 0.5$ thì quay lại bước lặp.

Bước 3.2: Nếu $\varepsilon_t = 0$ thì đây không phải là một phân lớp yếu. Khi đó nên tăng thêm số lượng mẫu huấn luyện.

Bước 4: Tính toán trọng số tốt α_t của h_t :
$$\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$$

Bước 5: Cập nhật lại trọng số

$$Với q_i = \begin{cases} e^{-\varepsilon_t} & \text{nếu } h_t(x_i) = y_i \\ e^{\varepsilon_t} & \text{nếu } h_t(x_i) \neq y_i \end{cases}$$

$$Và q_i \text{ mới: } q_i = \frac{W_i q_i}{Z_t} = \frac{W_i e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

Trong đó Z_t là thừa số chuẩn hóa sao cho $\sum_i W_i = 1$

Kết thúc (Termination):

$$f_T(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

Tính chất quan trọng nhất của thuật toán Adaboost là có f_{t+1} tốt hơn f_t .

Để đánh giá hiệu quả của thuật toán ta định nghĩa hàm sai số sau:

$$wrong(f, x_i, y_i) = \begin{cases} 1 & \text{nếu } f \text{ không phân lớp } x_i \text{ là } y_i \\ 0 & \text{nếu } f \text{ phân lớp } x_i \text{ là } y_i \end{cases}$$

Khi đó tổng sai số f_T trên N mẫu là:

$$E_T = \frac{\sum_{i=1}^N wrong(f_T, x_i, y_i)}{N}$$

3. Phương pháp phân tích thành phần chính (PCA-Principle Component Analysis)

Thuật toán PCA được trình bày lần lượt theo các bước dưới đây [7]:

Bước 1: Chuyển đổi ảnh

Biểu diễn M ảnh trong không gian 2D thành 1D. Tạo vector có kích thước N (số hàng của ảnh xám x số cột của ảnh xám) như mô tả sau:

$$\vec{S_i} = [a_1 \ a_2 \ \dots \ a_n]^T, i = \overline{1, M} \quad (3)$$

Với: a_i là giá trị pixel của ảnh. T là chuyển vị của ma trận S_i

Kết quả chuyển cho M ảnh đưa vào ma trận như sau:

Mỗi ảnh là một ma trận cột, ghép M ma trận cột ứng với M ảnh thành một ma trận có kích thước $N \times M$

$$P_{N \times M} = \begin{pmatrix} a_{11} + a_{12} + a_{13} + \dots + a_{1M} \\ a_{21} + a_{22} + a_{23} + \dots + a_{2M} \\ a_{31} + a_{32} + a_{33} + \dots + a_{3M} \\ \dots \\ a_{N1} + a_{N2} + a_{N3} + \dots + a_{NM} \end{pmatrix} \quad (4)$$

Chỉ số đầu là thành phần của vector, chỉ số sau là thứ tự của ảnh.

Bước 2: Tính ảnh trung bình

$$\vec{m} = \frac{1}{M} \sum_{i=1}^M x_i, i = \overline{1, M} \quad (5)$$

Chi tiết công thức trên:

$$\vec{m} = \frac{1}{M} \begin{pmatrix} a_{11} + a_{12} + a_{13} + \dots + a_{1M} \\ a_{21} + a_{22} + a_{23} + \dots + a_{2M} \\ a_{31} + a_{32} + a_{33} + \dots + a_{3M} \\ \dots \\ a_{N1} + a_{N2} + a_{N3} + \dots + a_{NM} \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ \dots \\ m_N \end{pmatrix} \quad (6)$$

Bước 3: Trừ mỗi ảnh cho ảnh trung bình

Nhằm mục đích tạo ra sự co giãn tương đối giá trị pixel của các ảnh.

$$\overrightarrow{n_{1m}} = \begin{pmatrix} a_{11} - m_1 \\ a_{21} - m_2 \\ a_{31} - m_3 \\ \dots \\ a_{N1} - m_N \end{pmatrix}, \overrightarrow{n_{2m}} = \begin{pmatrix} a_{12} - m_1 \\ a_{22} - m_2 \\ a_{32} - m_3 \\ \dots \\ a_{N2} - m_N \end{pmatrix}, \dots, \overrightarrow{n_{Mm}} = \begin{pmatrix} a_{1M} - m_1 \\ a_{2M} - m_2 \\ a_{3M} - m_3 \\ \dots \\ a_{NM} - m_N \end{pmatrix} \quad (7)$$

Xây dựng ma trận từ các $\overrightarrow{n_{im}}$ vừa tìm được

Đặt: $A = (\overrightarrow{n_{1m}} \overrightarrow{n_{2m}} \overrightarrow{n_{3m}} \dots \overrightarrow{n_{Mm}})$ sẽ được ma trận kích thước $N \times M$

Bước 4: Xây dựng ma trận Covariance

Nhằm mục đích thể hiện sự tương quan của từng vector đối với các vector còn lại trong không gian.

$$cov = AA^T \quad (8)$$

Tính trị riêng (*eigenvalue*: λ_i) và vector riêng (*eigenvector*: x_i) của ma trận Covariance này. Đó chính là thành phần đặc trưng thành phần thiết yếu của ảnh.

Trong thực tế, giả sử tồn tại một ảnh có kích thước 200x230 (*độ rộng và độ cao của ảnh*) thì khi đó kích thước của ma trận cov là 46000x46000 (N^2). Kích thước quá lớn do đó không thể tính trực tiếp trị riêng và vector riêng theo cách này. Vì vậy áp dụng lý thuyết đại số tuyến tính: λ_i, x_i có thể tính bằng cách giải quyết trị riêng, vector riêng của ma trận $A^T A$ (*kích thước $M \times M$ nhỏ hơn nhiều so với $N \times N$*).

Đặt μ_i và d_i là các trị riêng và vector riêng của ma trận $A^T A$. Kết quả như sau:

$$A^T A d_i = \mu_i d_i \quad (9)$$

Nhân mỗi vế của (9) cho A sẽ được:

$$AA^T (A d_i) = \mu_i (A d_i) \text{ với } X = A d_i \quad (10)$$

Điều này cho thấy: M vector riêng của x_i và M trị riêng của λ_i đầu tiên AA^T tương ứng chính là tích (A với vector riêng d_i của AA^T) và μ_i .

Các vector riêng là không gian đặc trưng của các biến báo trong cơ sở dữ liệu ảnh ban đầu. Các vector riêng được sắp xếp theo thứ tự từ cao đến thấp theo trị riêng tương ứng. Vector riêng có trị riêng càng cao sẽ mang nhiều đặc trưng thiết yếu nhất trong không gian các biến báo. Ở đây chỉ với M hướng đặc trưng mang giá trị riêng lớn nhất trong không gian $N \times N$ không gian đặc trưng.

Bước 5: Phép chiếu

Chiếu lần lượt các ảnh trong cơ sở dữ liệu đến không gian đặc trưng M , để sinh ra các biến báo đặc trưng trong không gian mới này.

$$\Omega_i = [x_1 x_2 x_3 \dots x_M]^T \cdot \overrightarrow{n_{im}} \quad (11)$$

Với $i = \overline{1, M}$, $[X]^T = [\overrightarrow{x_1} \overrightarrow{x_2} \overrightarrow{x_3} \dots \overrightarrow{x_M}]^T$ là các ma trận đặc trưng các biến báo đã rút trích ra được (gọi là các *eigensignal*). $\overrightarrow{n_{im}}$ là vector ảnh thứ i trừ đi ảnh trung bình.

Bước 6: Ảnh cần nhận dạng

Chuyển đổi ảnh cần nhận dạng thành vector 1 chiều: $\vec{r} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \dots \\ r_N \end{pmatrix}$ (12)

Tính sự sai số của ảnh cần nhận dạng với ảnh trung bình của các ảnh trong cơ sở dữ liệu.

Được vector sai số sau:

$$\vec{r} = \begin{pmatrix} r_1 - m_1 \\ r_2 - m_2 \\ r_3 - m_3 \\ \dots \\ r_N - m_N \end{pmatrix} \quad (13)$$

Chiều sai số này lên không gian đặc trưng của các biến báo.

$$\Omega_r = [\vec{x_1} \vec{x_2} \vec{x_3} \dots \vec{x_M}]^T \cdot \vec{r_m} \quad (14)$$

Bước 7: Nhận dạng biến báo

Như vậy đặc trưng của ảnh cần nhận dạng Ω_r và đặc trưng các biến báo trong cơ sở dữ liệu Ω_i đã được xây dựng. Phân loại biến báo bằng cách đơn giản nhất là dùng khoảng cách Euclide.

$$\epsilon_i = \|\Omega_r - \Omega_i\| \quad (15)$$

4. Thuật toán phân lớp Support Vector Machine (SVM)

4.1. Phân lớp tuyến tính (Linear classifier)

Ý tưởng của thuật toán SVM là xây dựng một mô hình để phân loại một đối tượng có thuộc hay không thuộc vào nhóm đối tượng cần quan tâm. Thuật toán SVM sẽ biểu diễn các điểm trong không gian và xác định ranh giới giữa hai nhóm đối tượng sao cho khoảng cách giữa tập dữ liệu được huấn luyện tới ranh giới là xa nhất có thể.

Tiến hành xét một bài toán đơn giản là tách hai lớp với tập dữ liệu mẫu đã được huấn luyện. Và sau đó ta có thể mở rộng phương pháp cho nhiều trường hợp tổng quát mà dữ liệu thậm chí không thể tách được phân lớp.

Với $x_i, i = 1, 2, \dots, N$ là tập các vector đặc trưng của bộ huấn luyện X. Và nó sẽ thuộc về một trong hai lớp w_1, w_2 và được giả sử rằng tập dữ liệu sẽ được phân lớp tuyến tính. Với mục tiêu là sẽ xây dựng mặt siêu phẳng (*hyperplane*) để tách chính xác các phân lớp mẫu được huấn luyện được cho bởi phương trình sau:

$$g(x) = \mathbf{w}^T \mathbf{x} + w_0 = 0 \quad (16)$$

Trong đó \mathbf{w} là vector trọng số, w_0 là độ dịch.

Với phương trình (16) ta sẽ xác định được mặt siêu phẳng. Mặt siêu phẳng giúp dễ dàng tách được hai phân lớp w_1, w_2 . Tuy nhiên trong thực tế có thể có nhiều hơn một mặt siêu

phẳng. Trong trường hợp này thuật toán SVM sẽ xác định mặt siêu phẳng dùng để tách phân lớp dựa theo khoảng cách cực đại giữa hai mẫu dữ liệu đã được huấn luyện. Và khoảng cách cực đại này còn được gọi là *lề (margin)*, mặt siêu phẳng này còn được gọi là *mặt siêu phẳng lề tối đa*.

Độ lớn của lề (margin) được cho như sau:

$$\frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|} \quad (17)$$

Từ phương trình (16) khi thay đổi w và w_0 hướng và khoảng cách từ gốc tọa độ tới mặt siêu phẳng. Bộ phân loại SVM được định nghĩa như sau:

$$f(x) = \text{sign}(w^T x + w_0) \quad (18)$$

Nếu $f(x) = +1$ thì x thuộc về phân lớp đang cần quan tâm, và ngược lại nếu $f(x) = -1$ thì x thuộc về lớp khác.

Phương pháp máy học SVM là tập các mặt siêu phẳng phụ thuộc vào các tham số w và w_0 . Mục tiêu của phương pháp SVM là ước lượng hai giá trị này để có thể cực đại hóa *lề (margin)*. Với giá trị của *lề* càng lớn thì mặt siêu phẳng phân lớp càng tốt.

Nếu tập dữ liệu huấn luyện là *khả tách tuyến tính* ta có các ràng buộc sau:

$$w^T x_i + w_0 \geq +1 \text{ nếu } y_i = +1 \quad (19)$$

$$w^T x_i + w_0 \leq -1 \text{ nếu } y_i = -1 \quad (20)$$

Hai mặt siêu phẳng có phương trình $w^T x_i + w_0 = \pm 1$ được gọi là mặt siêu phẳng hỗ trợ.

Phương pháp có thể tìm được giá trị w và w_0 để xây dựng được mặt siêu phẳng lề tối ưu là phải giải bài toán tối ưu toàn phương (Quadratic Programming).

$$\text{Cực đại hóa: } \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j x_i^T x_j \quad (21)$$

Với các ràng buộc sau:

$$\alpha_i \geq 0 \quad (22)$$

$$\sum_{i=1}^N \alpha_i y_i = 0 \quad (23)$$

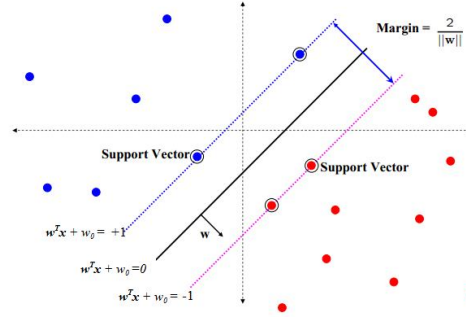
Trong đó các hệ số Lagrange $\alpha_i, i = 1, 2, \dots, N$ là các biến số cần phải tối ưu hóa.

Vector w sẽ được tính từ các nghiệm của bài toán toàn phương như sau:

$$w = \sum_{i=1}^N \alpha_i y_i x_i \quad (24)$$

Để xác định độ dịch w_0 , ta chọn x_i sao cho $\alpha_i \geq 0$, sau đó sử dụng điều kiện Karush-Kuhn-Tucker (KKT) như sau: $\alpha_i [y_i (w^T x_i + w_0) - 1] = 0$ (25)

Với $\alpha_i > 0$ là những mẫu nằm gần mặt siêu phẳng nhất và được gọi là các vector hỗ trợ (Support Vector). Theo hình 4 những mẫu được gọi là Support Vector là các mẫu xanh hoặc đỏ được khoanh tròn và nằm trên mặt siêu phẳng hỗ trợ (do dấu của bất đẳng thức bằng +1 nếu đây là mẫu cần quan tâm và bằng -1 với các mẫu còn lại).



Hình 4. Minh họa các mẫu được gọi là Support Vector.

4.2. Phân lớp phi tuyến (Nonlinear classifier)

Thực tế trong trường hợp tổng quát, mặt phân hoạch có thể là một mặt phi tuyến bất kỳ. Và ta chỉ cần *ánh xạ* vector dữ liệu vào không gian đặc trưng có số chiều cao hơn nhiều.

Giả sử các mẫu x_i thuộc không gian \mathbf{R}^n , không gian này được gọi là không gian giả thiết (*hypothesis space*). Để tìm mặt phi tuyến trong không gian này, ta ánh xạ các vector mẫu x_i từ \mathbf{R}^n vào một không gian \mathbf{R}^d có số chiều lớn hơn ($d > n, d$ có thể bằng ∞). \mathbf{R}^d được gọi là không gian đặc trưng (*feature space*). Sau đó áp dụng phương pháp SVM tuyến tính để tìm ra một siêu phẳng phân hoạch trong không gian đặc trưng \mathbf{R}^d . Siêu phẳng này sẽ là ứng với mặt phi tuyến trong không gian \mathbf{R}^d [6].

Trong không gian đặc trưng (*feature space*) này, các điểm dữ liệu trở thành *khả tách tuyến tính*, hoặc có thể phân tích với ít lỗi hơn so với trường hợp sử dụng không gian ban đầu. Khi đó, bài toán quy hoạch toàn phương ban đầu trở thành như sau:

Cực đại hóa:

$$\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad (26)$$

với các ràng buộc sau:

$$0 \leq \alpha_i \leq C \quad (27)$$

$$\sum_{i=1}^N \alpha_i y_i = 0 \quad (28)$$





















Trong đó k là một hàm nhân (*kernel function*) thỏa mãn:

$$k(x_i, x_j) = \Phi(x_i)^T \cdot \Phi(x_j) \quad (29)$$

Với việc sử dụng hàm nhân (*kernel function*), thì ta không cần quan tâm về ánh xạ Φ . Bằng cách chọn một hàm nhân phù hợp, ta sẽ xây dựng được nhiều bộ phân loại khác nhau. Ví dụ, chọn nhân đa thức $k(x_i, x_j) = (x_i^T x_j + 1)^p$ dẫn đến bộ phân loại đa thức, nhân Gaussian $k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$ dẫn đến bộ phân loại RBF (Radial Basis Functions) [6].

IV. KẾT QUẢ THỬ NGHIỆM

Bảng 1. Kết quả nhận dạng biển báo giao thông

Biển báo	 201a	 201b	 202	 205e	 207a	 207b	 207c	 209	 210	 225
 201a	98,50	0,50					1,00			
 201b	1,30	97,30				1,40				
 202			99,40	0,40	0,20					
 205e				90,50	9,50					
 207a				5,67	84,65	4,71	4,97			
 207b					8,54	83,35	8,11			
 207c					8,24	6,01	85,75			
 209						1,50		96,50		2,00
 210									97,50	2,50
 225			0,75						3,00	96,25
Độ chính xác trung bình: 92, 97%										

Từ các môi trường thử nghiệm ban ngày, ban đêm, trời có mây, mưa nhỏ, biển báo bị che khuất từ 10-15% diện tích, biển báo cũ, bong tróc sơn, bụi bẩn, biển báo bị nghiêng dưới 1 góc 20^0 , tốc độ di chuyển của phương tiện lưu thông từ 30-40km/hnhóm tiến hành đánh giá việc sai số giữa các biển báo giao thông được thể hiện ở bảng 1. Tập thử nghiệm gồm hơn 1500 ảnh tĩnh và 750 video clip.

Căn cứ theo bảng 1, chúng tôi nhận thấy tỉ lệ nhận dạng chính xác trung bình là 92,97%. Trong đó với nhóm các biển báo 201a, 201b, 202, 209, 210, 225 luôn cho tỉ lệ chính xác là $\geq 96\%$ bởi vì tính đặc trưng của các biển báo này là riêng biệt khó có thể nhầm lẫn sang các biển khác. Đối với nhóm các biển báo 207a, 207b, 207c có tỉ lệ nhận dạng chính xác là $\geq 83\%$, giữa các biển báo này dễ bị nhận dạng nhầm sang biển khác, vì thông tin đặc trưng riêng biệt giữa các biển là

khá giống nhau. Sự sai lệch này bị ảnh hưởng bởi nhiều yếu tố như: hình ảnh thu về có chất lượng kém, bị nhòe, hoặc do biển báo bị phai mờ, bị mất nét, nét vẽ quá mảnh không rõ, biển báo bị dính sơn, bụi bẩn, bị che khuất làm thông tin biển báo thu về bị sai lệch.

V. KẾT LUẬN

Trong bài báo này, chúng tôi đã tiến hành thực nghiệm đối với phương pháp được chọn sử dụng. Chương trình có khả năng phát hiện và nhận dạng chính xác thông tin các loại biển báo nguy hiểm được học với tỉ lệ trung bình 92,97%, hoạt động được trong nhiều loại môi trường phức tạp như đã trình bày ở trên, đáp ứng được nhu cầu xử lý trong thời gian thực.

TÀI LIỆU THAM KHẢO

- [1] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, T. Koehler, “A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information”, Proceedings. IEEE Intelligent Vehicles Symposium, 2005.
- [2] C. Y. Yang, C. S. Fuh, S. W. Chen, P. S. Yen, “A Road Sign Recognition System Based on Dynamic Visual Model”, CVPR’03 Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition, Pages 750-755, 2003.
- [3] Lê Thanh Tâm, Trần Thái Sơn, Seichii Mita, “Phát hiện và phân loại biển báo giao thông dựa trên SVM trong thời gian thực”, Tuyển tập Công trình Nghiên cứu Công nghệ Thông tin và Truyền thông, 2009.
- [4] Luis David Lopez and Olac Fuentes, “Color-Based Road Sign Detection and Tracking”, International Conference on Image Analysis and Recognition (ICIAR), Montreal, CA, August 2007.
- [5] Nguyễn Duy Khánh, Lê Đình Duy, Dương Anh Đức, “Phát hiện biển báo giao thông dùng đặc trưng cục bộ (local features)”, Hội thảo FAIR (Fundamental And Applied IT Research) lần V, Tháng 08-2011.
- [6] Nguyễn Linh Giang, Nguyễn Mạnh Hiền, “Phân loại văn bản tiếng Việt với bộ phân loại vector hỗ trợ SVM”, Tạp chí CNTT&TT, Tháng 06 năm 2006.
- [7] Phan Sau Ra, Đặng Thị Kim Yến, “Face Recognition Using PCA And Neural Network”, Đồ án tốt nghiệp, Đại Học Sư Phạm Kỹ Thuật TP.HCM, 2012.
- [8] Viola and Jones, “Rapid object detection using a boosted cascade of simple feature”, Computer Vision and Pattern Recognition, 2001.
- [9] “Adaptive Boosting – Adaboosting”, <http://www.csie.ntu.edu.tw/~b92109/course/Machine%20Learning/>
- [10] “Đánh giá tình hình trật tự An toàn giao thông”, <http://www.mt.gov.vn/Default.aspx?tabid=26&catid=204&articleid=12826>