視覚型道路標識検出システムの構築

15300051

平成 15 年度～平成 17 年度科学研究費補助金

（ 基盤研究（B） ）研究成果報告書

平成 18 年 4 月

研究代表者　平井有三

筑波大学 大学院システム情報工学研究科教授

「はしがき」

平成15年に国内で発生した交通事故件数は約95万件で、7,700人が死亡し、118万人が負傷している。発生件数の約6割を追突事故と出会い頭の事故が占めている。米国でも、2002年に680万件の事故で42,000人が亡くなっている。このような背景の下、世界でITS(Intelligent Transport System)に関する研究が1990年頃から盛んに行われてきた。国内では、高度道路交通システム(ITS)の一環として、自動車運転の安全と円滑さを目標に走行支援道路システム(AHS: Advanced cruise-assist Highway System)に関する研究が、多くの研究機関で行われている。

AHSには、AHS-i、AHS-c、AHS-aの三つのレベルがあるとされている。AHS-iは情報収集の一部をシステムがサポートするものであり、前方障害物情報の表示板による提供など、現在実用化がもっとも進んでいるレベルである。AHS-cは運転操作の一部をシステムがサポートするレベルであり、AHS-aは完全自動運転のレベルである。

AHS-iにおける情報収集にもいろいろな対象が考えられる。現在実用化されている前方障害物情報は数キロメートルのレンジであり、運転者からは直接見えない情報である。しかしながら、平成15年の事故統計によれば、安全不確認、脇見運転、動静不注視などの事故原因が、全体の55%を占めている。したがって、運転者から直接見える情報を見落とさないように支援することも重要な研究課題である。

本研究の目的は、運転者から直接見える情報、特に道路高越標識の見落とし防止を支援する技術を開発することにある。道路交通標識は、運転者にとってもっとも身近な見える対象であり、かつその見落としが直接大きな事故につながる危険性がある。本研究で識別対象とした道路交通標識は、赤い円環状の輪郭を持った、速度規制（30，40，50ｋｍ）、追い越し禁止、駐車、駐停車禁止の6種の標識である。現在、約50km／時で走行している車載ビデオ記録映像から、10フレーム／秒の速度で、約97%の認識精度で標識を認識できている。

「研究組織」

研究代表者：平井有三（筑波大学 大学院システム情報工学研究科・教授）

研究分担者：酒井 宏 （筑波大学 大学院システム情報工学研究科・助教授）

「交付決定額 （配分額）」 （金額単位：円）

	直接経費	間接経費	合計
平成15年度	3,700,000	0	3,700,000
平成16年度	9,500,000	0	9,500,000
平成17年度	2,800,000	0	2,800,000
総計	16,000,000	0	16,000,000

「研究発表」

(1) 国際会議

(ア) Yutaka Ishizuka and Yuzo Hirai: "Segmentation of road sign symbols using opponent-color filters." Proceedings of ITS World Congress 2004, 8 pages in CD-ROM, 18-22, October 2004

(2) 口頭発表

(ア) 石塚 裕、平井有三：「色情報を用いた実時間道路交通標識の切り出し」電子情報通信学会技術研究報告、ITS-2002-66,IE-2002-207、pp.139-144、2003年2月

(イ) 石塚 裕、平井有三：「Opponent-Colorフィルタを用いた道路交通標識認識システム」電子情報通信学会技術研究報告、PRMU2003-258、pp.13-18, 2004年3月

(ウ) 馬場今日子、平井有三：「車載カメラ画像による道路交通標識のリアルタイム認識」電子情報通信学会技術研究報告、ITS2004-48、IE2004-182、pp.45-50、2005年2月

(エ) 馬場今日子、平井有三：「反対色フィルタを用いた道路交通標識の実時間認識」画像の認識・理解シンポジウム（MITRU2005）講演予稿集、pp.939-946、2005年7月

(オ) 馬場今日子、平井有三：「反対色フィルタによる色画像のセグメンテーション−道路交通標識の実時間認識への応用−」日本神経回路学会全国大会予稿集、2 pages、2005年9月

(カ) 馬場今日子、平井有三：「反対色型視覚フィルタによる道路交通標識の

実時間認識」第4回ITSシンポジウム、pp.325-330、2005年12月

「研究成果による工業所有権の出願・取得状況」

なし

「研究成果」

次ページ以降に、上記研究発表の内容を順に報告する。また、本研究で開発した画像処理アルゴリズム開発用システムについて

1. ImageFilterPlugin仕様書
2. 画像操作クラス：CDIB仕様書

を附属資料として添付する。

また、平成16年度に開発した、ビデオカメラから実時間で道路画像を取り込み道路交通標識の認識処理を行うための、

3. 実時間認識処理ソフトウエア

のプログラムを添付する。

Presented at ITSWC2004 Nagoya
18-22, October 2004

SEGMENTATION of ROAD SIGN SYMBOLS using OPPONENT-COLOR FILTERS

Yutaka Ishizuka
Master's Program in Science and Engineering, University of Tsukuba
Tennodai 1-1-1 Tsukuba, Ibaraki 305-8573, Japan
yishizuka@viplab.is.tsukuba.ac.jp

Yuzo Hirai
Institute of Information Sciences and Electronics, University of Tsukuba
Tennodai 1-1-1 Tsukuba, Ibaraki 305-8573, Japan
hirai@is.tsukuba.ac.jp

ABSTRACT

A novel algorithm for the recognition of Japanese road sign symbols with red annular boundaries is proposed. After detecting a red annular object using HSV color coordinates, a biologically inspired opponent-color filter is applied to extract the symbol parts of road signs. By applying discriminant analysis to the filter output, the inside region circumscribed by red annular boundary can be extracted. By applying the discriminant analysis again to the inside region, the symbol part of road sign can be segmented. The extracted symbol is classified by a decision tree and recognition rate of about 96% was achieved for real road video images. It is shown that the opponent-color filter can lead to accurate recognition even for small and degraded road signs in video images.

INTRODUCTION

Automatic recognition of road signs for the assistance of safety driving is one of the most important components in the intelligent transportation system (ITS). Although many vision systems for the automatic recognition of road signs have been proposed in the last decades[5],[3], they are still at the experimental level because of many difficulties in the task: Variations in lighting conditions, weather conditions, motion and/or focus blurring, degeneration of paints, etc. One way to challenge these difficulties is to learn how our visual system solves these problems, since our vision is the most reliable and robust system in the world. In this paper, as the first step towards a biologically inspired road sign recognition system, opponent-color filters found in our visual systems are applied to the segmentation of symbols in road signs and it is shown that the filters can extract symbols reliably.

This paper first describes algorithms for the segmentation, extraction and classification of road signs. Then, the performance of the system measured by real video images will be discussed and analysed.

VISION SYSTEM

In this section, video camera and PC, algorithms for the segmentation, extraction and classification of road sign will be described.

VIDEO CAMERA AND THE SYSTEM

The traffic images analyzed in this paper have been acquired by a video camera (SONY DCR-TRV900 NTSC) mounted on a tripod inside a car. Road signs not only stand on the left or the right side of road, but also hang over lane. Overhead signs are common and even hang over opposite lane as shown in Figure 1(a). Since both overhead and side signs are our targets, the focus of video camera is set at the widest angle (f=4.3mm), so that each image of road sign becomes inevitably small (from 20 to 60 pixel wide). Video images are captured through an IEEE 1394 DV terminal on PC (Dell Precision 340, 2.8GHz, 2GB memory), and off-line but video-rate processing has been done on the PC.

(a)

(b)

Figure 1: (a) A scene of overhead and side signs. (b) Nine road signs used in this paper. From left to right and from top to bottom row: Speed limits to "30Km", "40Km" and "50Km", "No Parking", "No Parking and Standing", "No Passing", "Close to Traffic", "No Throughfare" and "No U-turn".

DETECTION OF ROAD SIGNS

In this paper we focus on nine road signs shown in Figure 1(b) whose boundaries are composed of red annuli, and try to thoroughly investigate difficulties in this task. Extension to other types of road signs may be less difficult than this task, since most of the difficulties encountered in this task will be common to the other signs. The process of road sign recognition consists of three stages: (1) Detection of red circular objects, (2) segmentation of symbols in the objects and (3) classification of the symbols.

Detection of red objects

Detection of red circular objects is carried out first by segmenting red parts in a scene according to HSV color coordinates. When a color is defined by RGB color coordinates as (R, G, B), where R, G and B are between 0.0 and 1.0, and let MAX $= \max\{R, G, B\}$ and MIN $= \min\{R, G, B\}$, the transformation from RGB to HSV coordinates is given by

$$H = \begin{cases} (0 + \frac{G-B}{\text{MAX}-\text{MIN}}) \times 60 & \text{if } R = \max, \\ (2 + \frac{B-R}{\text{MAX}-\text{MIN}}) \times 60 & \text{if } G = \max, \\ (4 + \frac{R-G}{\text{MAX}-\text{MIN}}) \times 60 & \text{if } B = \max, \end{cases} \quad (1)$$

$$S = \frac{\text{MAX} - \text{MIN}}{\text{MAX}}, \tag{2}$$

$$V = \text{MAX}, \tag{3}$$

where H varies from 0.0 degree to 360.0 degree and 360.0 wraps around to 0.0, and S and V vary from 0.0 to 1.0.

Each pixel is classified as red when the hue is from 324 degree to 28.8 degree, the saturation is above 0.2, and the value is above 0.1. An example of road scene image is shown in Figure 2(a) and the result of red-labeling is shown in (b).

(a) (b) (c)

Figure 2: (a) A scene of road signs. (b) Labeling red pixels. (c) Segmentation of red annular object as a candidate for road sign.

Detection of red annular objects

In order to determine whether each red object is an annulus, first the outer edge of a red object is traced and a circle equation is obtained by three points on the edge as follows. As illustrated in Figure 3(a), the equation of circle which passes through the origin is given by

$$x^2 + y^2 - 2gx - 2fy = 0 \tag{4}$$

where (g, f) is a set of coordinates of the center of a circle. Although the number of parameters of a circle is three, one of them, i.e. the radius r, is given by

$$r = \sqrt{g^2 + f^2}, \tag{5}$$

since the circle is constrained to pass the origin. Given two other points on the circle, two other parameters, i.e. the center of the circle (g, f), can be obtained by solving the following simultaneous equations

$$\begin{cases} x_1^2 + y_1^2 - 2gx_1 - 2fy_1 = 0 \\ x_2^2 + y_2^2 - 2gx_2 - 2fy_2 = 0. \end{cases} \tag{6}$$

It may be natural to choose the three points from equally spaced pixels along the traced edge.

Since the circle equation is derived from only three points, it is necessary to check if the other edge pixels match the circle. Let the length between each edge pixel and the center of the circle (g, f) as q_i, $i = 1, \ldots, N$, where N is the number of edge pixels. Error is defined by

$$E = \frac{1}{N} \sum_{i}^{N} \left(1 - \frac{r}{q_i}\right)^2 \tag{7}$$

where r is the radius of the circle. Red object is judged as a red annular object when the error E is less than some threshold. The threshold was set to 0.015 in this paper. We do not care the

inner edge of an annulus because if the red object is not annulus, it is not a road sign and it will not be classified as a road sign by the recognition stage.

Figure 3(b) shows a sample image of red objects and (c) shows a traced edge and a circle obtained by the three-point algorithm. As seen in the figure, this algorithm seems to work well. Red annular object is segmented by circumscribing the circle with a rectangle as shown in Figure 3(c) and Figure 2(c).

Since the hue in HSV coordinates is stable against variations in lighting conditions, detection of red circular road signs can be carried out in a robust manner, if images do not suffer from large motion blurs, as previous works have suggested [3].

(a) (b) (c)

Figure 3: (a) Definition of a circle equation. (b) Sample image of red objects. (c) Traced edge boundary and extracted circle by the three-point algorithm. The result of segmentation is shown in Figure 2(c).

SEGMENTATION OF SYMBOLS

After red annular objects are segmented, they are classified as one of the nine kinds of road signs or different from them. Most of the algorithms hitherto proposed, e.g. [5], [3], have used template matching between the segmented image and template patterns of road signs. Since distance between a camera and a road sign will change as car moves, size of the segmented road sign will change and it must be normalized to match the size of templates. Since lighting condition will also change, brightness and color of the segmented object should also be normalized.

Even for the same road signs, however, shape and position of the symbols may be different from manufacturer to manufacturer or from town to town. In order to absorb these variations, a large number of templates for each road sign will be necessary.

An alternative method, which may be more robust to the variations in position and shape, is to use decision trees for structural pattern recognition [2]. In this method structural features, which are useful for classifying road signs, must be extracted from symbol images in the road signs. Segmentation of symbols, however, is not an easy task, since they are small and their quality may be poor because small symbol must be segmented from motion images.

Segmentation of symbol by color cue only may not be satisfactory, because hue in HSV color space is not stable in the neighborhood of value or gray-level axis where saturation is small. Small perturbations in RGB signals will cause large variations in hue. In order to suppress these variations, we tried smoothing filters such as bilateral filters [7] and mean shift algorithms [1], but they were not satisfactory for small and degraded images as shown below.

Opponent-color filter

Our visual systems have three kinds of photo receptors, L-, M- and S-cone, or R, G and B receptors[6]. Ganglion cells, that are output cells of retina, have several types of receptive fields. A typical cell, called an L-M opponent color cell, has a receptive field composed of excitatory and inhibitory concentric regions as shown in Figure 4(a). Center region receives excitatory inputs from L-cones and annular region receives inhibitory inputs from M-cones. It responds to small red light presented at the excitatory region, but the response is suppressed by presenting green light to the surrounding inhibitory region simultaneously. It does not respond to a large white or yellow light because they contain red and green components which induce excitation and inhibition simultaneously and their effects will be canceled out. It does not responds to blue light because it does not contain red and green components. For road signs shown in Figure 1(b), white or blue colors are used as background to emphasize interior symbols whose colors are blue or red, respectively. Therefore, opponent color filter will be suitable for detecting symbols from these color arrangements.

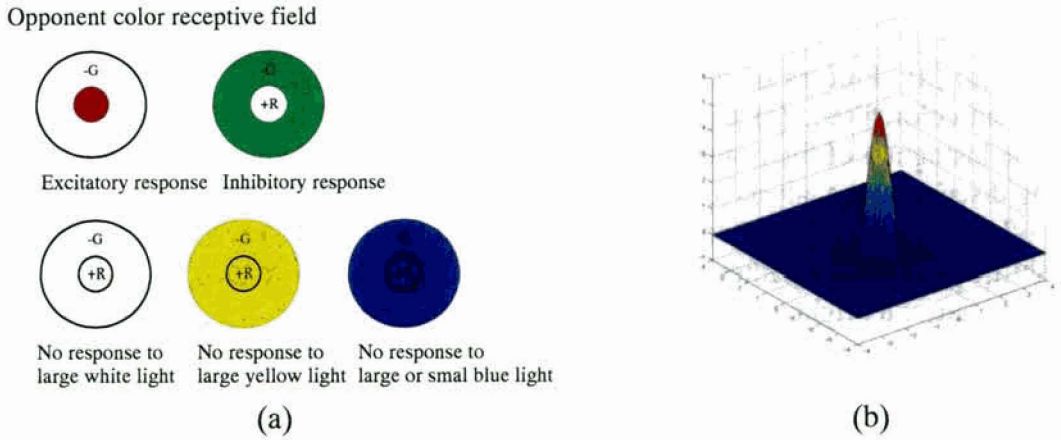


Figure 4: (a) Receptive field structure of a typical L-M opponent color cell and its responses to color stimuli. (b) Two-dimensional Laplacian- Gaussian function used for opponent color filter.

Opponent color filter is designed by a two-dimensional Laplacian-Gaussian function defined by

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{8}$$

$$F(x,y) = -\nabla^2 G(x,y) = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{9}$$

where $G(x,y)$ is a two-dimensional Gaussian function and the filter is derived by applying Laplacian operator to the Gaussian function. The three-dimensional structure of the filter is shown in Figure 4(b). Among (R,G,B) signals, R-signal is fed to the excitatory part of the filter and G-signal is fed to the inhibitory part. Parameter σ determines the spatial size of the filter and it is automatically adjusted according to the size of the detected annular region.

Segmentation of symbol is carried out in two stages. Opponent-color filter is applied to the original image, shown in Figure 5 (a), and the filter output is shown in Figure 5 (b). In order to segment symbol part, the filter output is binarized according to the amplitude of the output. Threshold for the binarization is automatically determined by discriminant analysis, that finds threshold maximizing the ratio of between-class variance and within-class variance of the amplitude histogram (e.g. [2]). By applying discriminant analysis to the amplitude histogram,

shown in Figure 6(a), of the filter output, central region circumscribed by a red annular frame of road sign can be extracted as shown in Figure 5(c). By applying discriminant analysis to the amplitude histogram, shown in Figure 6(b), of the central region again, symbol parts of road signs can be segmented from the background effectively as shown in Figure 5(d), even from degraded original image as illustrated in this example. This road sign was recognized correctly as speed limit to "30Km".

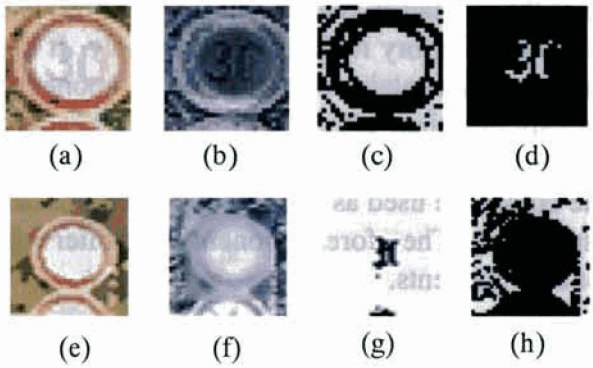


Figure 5: (a)-(d): Process of segmentation of symbols.(a) Original input image. (b) Output of opponent-color filter. (c) Digitized image obtained by applying discriminant analysis to amplitude histogram of filter output shown in Figure 6(a). (d) Segmented symbol obtained by applying discriminant analysis to the amplitude histogram of the central circle region shown in Figure 6(b). It was recognized correctly as speed limit to "30Km". (e) and (f) are the results of mean shift[1] and bilateral filtering [7], respectively. (g) and (h) are the results of discriminant analysis on hue and brightness distribution of the original image (a). As seen in the results, they are not satisfactory.

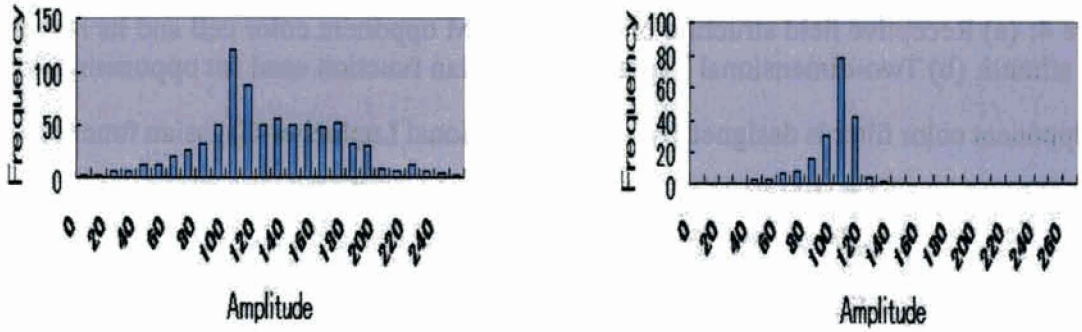
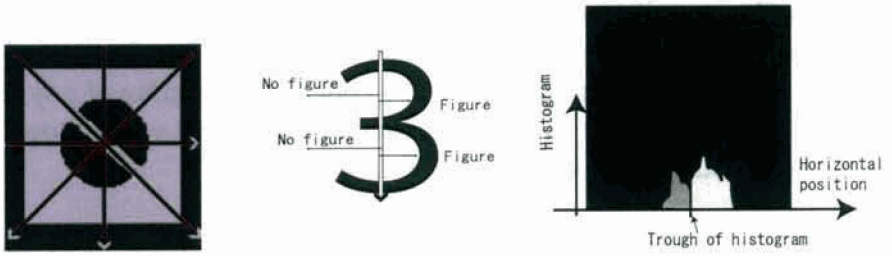


Figure 6: Amplitude histograms of opponent color filter output (a) and central region (b).

Results of segmentation by other methods are also shown in the figures from (e) to (h). Mean shift (e) [1] and bilateral filter (f) [7] are smoothing filters which take into account not only similarities of pixel properties such as color, but also neighborhood relationship in a two-dimensional image plane. As seen in the results, numerical symbols are smoothed out. We tried to segment original images in HSV color space because it may provide the simplest solution. Discriminant analysis was applied to hue (g) and brightness distributions (h), but the results are not satisfactory as seen in the figures.

IDENTIFICATION OF ROAD SIGNS

Symbols are classified by extracting structural features such as the number of pixels in four orientations, background features similar to those proposed in [4], and patterns of horizontal and vertical projection profiles as illustrated in Figure 7. Decision trees using these features are hand-coded for the classification. The target road signs are nine signs as shown in Figure 1(b).

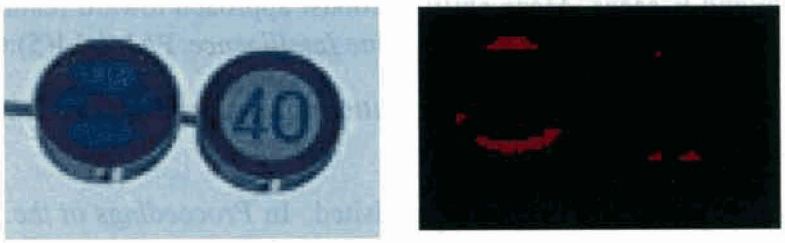


(a) (b) (c)

Figure 7: Examples of structural features. (a) "No Parking", "No Parking and Standing", Close to Traffic", "No Throughfare" and "No U-turn" are classified by pixel distributions in four orientations. (b) Numrical symbols composed of speed limit signs are classified by simplified background features [4]. (c) "No Passing"is classified by a projection profile.

PERFORMANCE EVALUATION

The performance of the system has been tested by traffic video images taken by ourselves. Detection rate of red annular road signs was about 76% and 137 road signs were detected. This relatively low score results from several factors. The most harmful one is lighting condition. When a overhead road sign, for example, is seen against bright sky background, red annular frame becomes black and detection by HSV color coordinates will corrupt as shown in Figure 8. In order to avoid this kind of failures, backlight compensation by brightness sensitivity control circuits or ND filter may be effective.



(a) (b)

Figure 8: An example of detection failure. When road signs are seen against bright sky as shown in (a), the hue of red annuli black out and they cannot be detected as shown in (b).

Another factor is motion blur. A trivial solution for this problem will be to increase the shutter speed of video camera at the expense of brightness of the image.

Among 137 road signs detected, 132 were correctly classified and the recognition rate was about 96%. Although the shape and the color of road signs used for the creation of decision rule are different from those of road signs actually taken in the video images, high recognition rate could be obtained. For each type of road sign, we used only one example of road sign for the creation of decision rules.

Since 584 frame images were detected as candidates of red road signs, each road sign has a sequence of about $4.3(=\frac{584}{137})$ frame images. Among 584 frame images, 67 were misclassified, so that about 0.5 frame images were misclassified for each road sign. Among 22 frame images that were not road signs but detected as the candidates, 8 images were misclassified as road signs, so that false alarm rate was about $1.3\%(=\frac{8}{584+22})$. False alarms may well be detected because they appear as solitary frames and do not appear in succession, on the other hand each road sign appears in a sequence of about 3.8 correct frames on average,

Processing time for the detection of three road signs in a image, for example, took about 120 msec by Pentium 4 with 2.8 GHz cpu and VC++ code. Segmentation of a symbol by opponent-color filter took less than 5 msec and classification of segmented symbol took less than 5 msec. Therefore, the processing speed was about 130 msec/frame.

CONCLUSIONS

We will continue the experiment in various different conditions to make the system more robust and accurate. In order to overcome backlight problem, edge should be detected by brightness contrast in addition to hue information as described in this paper. We are planning to improve circle fitting algorithm to unite broken annuli.

Passive probing such as vision system is ecologically gentle, but vision in general environment is a quit difficult task. To challenge this problem, learning our visual system will provide promising cues for that solution.

Acknowledgement

A part of this work has been supported by JSPS KAKENHI(15300051).

References

[1] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-14(5):603–619, 2002.

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition).* John Wiley & Sons, Inc., New York, 2001.

[3] D. M. Gavrila. Traffic sign recognition revisited. In *Proceedings of the 21st DAGM Symposium for Mustererkennung.*, pages 86–93. Springer Verlag, 1999.

[4] H. A. Glucksman. Classification of mixed font alphabetics by characteristic loci. In *Digest of 1st Ann. IEEE Computer Conference*, pages 137–141, 1967.

[5] G. Piccioli, E. De Micheli, P. Parodi, and M. Campani. Robust method for road sign detection and recognition. *Image and Vision Computing*, 14:209–223, 1996.

[6] L. Spillmann and J. S. Werner. *Visual Perception: The Neurophysioligical Foundation.* Academic Press, San Diego, California, 1990.

[7] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

ITS2002-66, IE2002-207, pp.139-144, 2003-02

信学技報
TECHNICAL REPORT OF IEICE

色情報を用いた実時間道路交通標識の切り出し

石塚 裕† 平井 有三‡

†筑波大学大学院理工学研究科 〒305-8573 つくば市天王台 1-1-1 筑波大学理工学研究科
‡筑波大学電子情報工学系 〒305-8597 つくば市天王台 1-1-1 筑波大学電子情報工学系

E-mail: † yishizuka@viplab.is.tsukuba.ac.jp ‡ hirai@is.tsukuba.ac.jp

あらまし 自動車に搭載されたカメラ映像から，道路標識を実時間で切り出すためのアルゴリズムを提案する．道路標識の色情報を用いたシステムを構成し，効率よく高速に道路標識の切り出しが行えることを示す．また同じアルゴリズムで，夜間の道路標識抽出も行えることを示す．

キーワード ITS，道路標識，実時間，切り出し，楕円あてはめ問題，ハフ変換

Real-time Extraction of Road Traffic Signs using Color Information

Yutaka ISHIZUKA† and Yuzo HIRAI‡

† Master Program of Science and Engineering, University of Tsukuba 1-1-1 Tennoudai,Tsukuba-shi,Ibaragi,305-8573 Japan
‡ Institute of Information Sciences and Electronics, University of Tsukuba 1-1-1 Tennoudai,Tsukuba-shi,Ibaragi,305-8597 Japan
E-mail: † yishizuka@viplab.is.tsukuba.ac.jp, ‡ hirai@is.tsukuba.ac.jp

Abstract We propose an algorithm that can extract road traffic signs in real-time using color information. We applied the algorithm to video images taken from a car and verified that the algorithm performed satisfactorily. It is shown that the same algorithm can extract road signs at night.

Keyword ITS, Road Traffic Sign, Real-time, Cutout, Ellipse, Hough Transform

1. はじめに

日本では，高度道路交通システム(ITS)[1]の一環として自動車運転の「安全と円滑さ」を目標にAHS(Advanced cruise-assist Highway System)の研究が多くの研究機関でなされている[2][5][6]．AHSは，運転を支援する機能に応じて以下の三種類のサブシステムに分類されている．

(a) AHS-i(情報収集)
(b) AHS-c(運転操作をサポート)
(c) AHS-a(完全自動運転)

AHSのうち初段階である，AHS-iを実現する手段としては，電磁波，レーダー波などから得られる情報をコンピュータを用いて認識するシステム等も考えられるが，本研究では，カメラ映像を用いたリアルタイム運転者支援システムの構築を目指している．

2. システム構成

ドライバーに，より安全で正確な情報を与えるために必要な技術に対しては，

1.高速に処理することができること
2.高い精度で認識することができること
3.取りこぼしなく抽出できること
4.周囲の環境の変化にロバストであること

が要求される。特に高速でかつ高い精度で認識ができることは、交通の安全上絶対不可欠な要素である．

以下ではこれらのために必要なシステムについて述べる

2.1. 対象標識

道路交通標識は，赤，青，黄，緑の色特徴と，円形，菱形，正方形，三角形といった形状の特徴を持っている．本研究では，主に赤い，円形の道路交通標識と，市街地でよく見られる，青，黄色の標識の切り出しを目的としている．本アルゴリズムを適用すれば，すべての円形な道路標識を切り出すことができる．

2.2. リアルタイム動画像処理システム

本研究では，静止画についてまず切り出しアルゴリズムを開発，有効性を検証し，動画像処理に応用していく．リアルタイムで道路標識を切り出し，認識して行くには，高速かつ正確なアルゴリズムが必要となる．以下の図1が本システムが目標とする構成である．

本システムは，4つの部分からなっている．カメラ

からの映像取り込み部，画像処理部，認識部，最後は追跡/ドライバーへの情報提供部になる．これらの処理を自動車の走行中に繰り返すことで，ドライバーに情報を提供する．本稿では，まず，デジタルスチルカメラで撮影された静止画に対して，切り出しアルゴリズムを適用し，アルゴリズムの有効性を確かめ，最後に，車載カメラより取り込んだ動画像でリアルタイム動画像処理を行い，システムを完成させる．

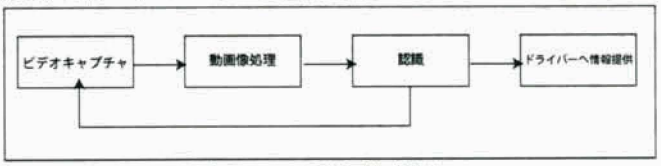


図 1 システム構成図

3. アルゴリズム

3.1. HSB 表色系

ビデオカメラから得られる画像は通常 RGB 表色系になっており，このままでは，色相情報を取り出すことができない．そこで，RGB 表色系から HSB 値を算出した．HSB とは，色相(Hue)，彩度（Saturation），明度(Brightness)のパラメータであり，RGB から相互に変換できる表色系の一つである．HSB 表色系を用いることで，色合いと鮮やかさ，明るさを別々のパラメータで表すことが出来るようになる．これにより，明るさに対してロバストな色情報を獲得することができる．

3.2. 色相による色のラベル付け

道路交通標識は，赤，青，黄，緑，白，黒の 6 色からなっている．そこで，ここでの色のラベル付けプロセスでは，入力画像の RGB 値を HSB 値に変換し，色相と彩度のパラメータを元にピクセルごとにラベル付けをする．ラベル付けには，R,G,B 値を用い，これらの組み合わせでラベルを表現するものとする．また付加する値は，0 か 255 かの 2 値である．こうすることで，後の画像処理が勘弁になる．たとえば，「赤」というラベルを付けたいピクセルに対しては，(R,G,B)=(255,0,0)を割り当てた．このようにして，ラベルを視覚的に表現した．(図 3)

本研究では，ラベル付けのための閾値を以下の表 1 のように設定した．また，今後はこの RGB 値でラベル付けされた画像に対して処理を行う．

表 1 ラベル付け対応表

ラベル	色相(0~1.0)	彩度(0~1.0)
赤(255,0,0)	0.08 以下，0.9 以上	0.2 以上
黄(255,0,255)	0.08 以上，0.2 以下	0.5 以上
緑(0,0,255)	0.2 以上，0.5 以下	0.7 以上
青(0,255,0)	0.5 以上，0.74 以下	0.5 以上
無し(0,0,0)	上記以外	上記以外

3.3. ノイズ除去

ラベル付けされた画像には，いくらかのノイズが残る．そこでこのノイズを除去するために，以下の図 2 の様な 3x3 のマトリックスを用いたフィルタリングを行う．着目しているピクセルを含め周囲 9 近傍にある同じラベルを持ったピクセルの個数を数える．この個数がある閾値 T よりも大きい場合は，着目しているピクセルに同じラベルを付ける．反対に小さい場合は，ノイズとして扱い，ラベルを取り除く(「黒」のラベルを付ける)．

このようにして，色相と彩度を元に，ラベル付けを行うことで，ある程度の道路交通標識を取り出すことができるようになる．

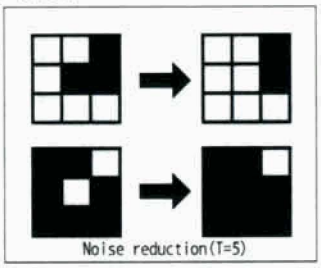


図 2 ノイズ除去

以下の図 3 は，入力画像をラベル付けし，ノイズを除去したものである．このような処理を施すだけでも道路交通標識部分だけを見ることができる．

図 3 ラベル付けとノイズ除去

3.4. 標識部分の切り出し

以下では，色のラベル付け，ノイズ除去が終わった画像から，道路交通標識であると思われる領域を切り出す．以下の節ではその原理を述べる．まず，ラベルのエッジをトレースし，その過程で円，楕円のパラメータを求める．現在対象としている道路交通標識は，主に赤い円形の道路交通標識であるので，円のパラメータだけを求めればよいのだが，標識を見る角度によっては，円形の標識も楕円に見えることがある．したがって円，楕円のパラメータを求め比較した．

トレース過程で円，楕円のパラメータを求めるのは，一般に，パラメータを求めるためには多くのエッジ点がある方が精度の向上を望めるからである．また，一般に，サンプリング誤差，量子化誤差の影響で，真の円，楕円のパラメータは一意に求まらず，いくつかのパラメータ候補が求められる．このパラメータの中から適切であると思われるパラメータを選ばなくてはならない．ここでは，そのためにハフ変換的な投票を行

い，もっともらしいパラメータを選び出すことにした．このもっとも多くの投票を得たパラメータから，円，楕円の外接矩形を算出し画像中より切り出す．また，切り出しの際には，若干の事前知識を導入した．

事前知識は以下のような物である．

1． 切り出した領域が大きすぎず，小さすぎないこと

2． 切り出した領域が，縦長や横長ではなく，適度な長方形であること

このような事前知識を導入することで検出率の向上と切り出し速度の向上を図っている．

3.4.1. エッジのトレース

ここでは，この段階までに得られたRGB値でラベル付けされた画像から道路標識を切り出す．ここまでの処理で得られた画像の中に含まれる閉じた領域をトレースすることで切り出しを行った．これにより，閉じた領域を個々に識別することができ，前段階で取り除けきれなかったノイズをさらに除去，最終的に標識を切り出すことができる．また平行して楕円を検出することもできる．以下がそのアルゴリズムである．

a）画像全体をラベルを持つピクセルが見つかるまで走査する．(トレース開始)

b）トレースを開始する点から，時計回りにエッジをトレースする．(トレースステップ)

c）トレースを開始した点に到着したら，トレース終了．これを一つのオブジェクトとして識別．（トレース終了）

d）トレースした領域を画像領域から取り除く

以上のステップを画像全体に対して行う．この際，領域の面積がある閾値よりも小さい場合はノイズとして取り除いた．

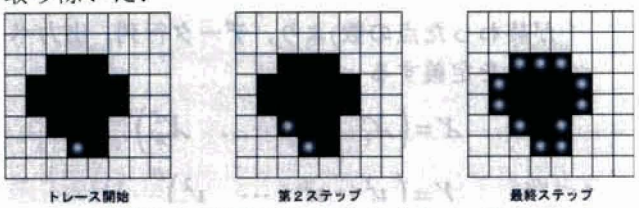


図4 オブジェクトのトレース

3.4.2. 円のあてはめ

以上が，領域切り出し手順であるが，円形な道路交通標識を切り出すためには，円かどうかを調べる必要がある．ここでは，エッジをトレースしている間，その閉じた領域が円であるかどうかを調査する．円を検出する流れは以下の通りである．

原点を通る円の方程式は以下のように与えられる．

$$x^2+y^2-2gx-2fy=0 \quad (1)$$

(g,f):円の中心座標

原点とその他2点，$P_1(x_1,y_1),P_2(x_2,y_2)$を通る円は，

$$\begin{cases} x_1^2+y_2^2-2gx_1-2fy_1=0 \\ x_2^2+y_2^2-2gx_2-2fy_2=0 \end{cases} \quad (2)$$

この連立一次方程式を解くことで，円の中心(g,f)を求めることができる．また，円の半径は，円が原点を通るという制約のため，半径は$r=\sqrt{g^2+f^2}$である．(図5)

トレースを開始する点を原点とする新たな座標軸を設定し，トレース中にエッジ点列より2点をランダムに選び出し，この方程式を解く．求まった(g,f)が円のパラメータとなる．

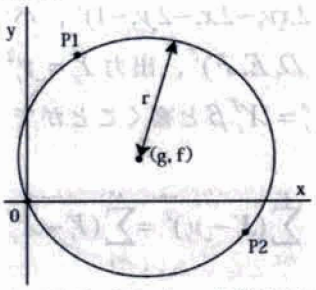


図5 円のパラメータ算出方法

円あてはめアルゴリズム

1． nが閉じた領域のエッジ点の総数Nになるまで以下2-5を繰り返す

2． トレース開始点が原点になるように平行移動されたエッジ点列$P=\{p_1,p_2,p_3,,,p_n\}$(n=トレースが終わった点の数)より，任意の2点$P_i(x_i,y_i),P_j(x_j,y_j)$を選ぶ．

3． 以下の連立一次方程式を解く．

$$\begin{cases} x_i^2+y_i^2-2gx_i-2fy_i=0 \\ x_j^2+y_j^2-2gx_j-2fy_j=0 \end{cases}$$

4． 求まった(g,f)を投票空間に投票する．

5． 次の点をトレースする．

6． 投票空間で最も投票を得たパラメータ(G,F)を円のパラメータとする．

3.4.3. 楕円のあてはめ

楕円も同様にしてあてはめを行うことができるが，楕円の場合はパラメータも多く複雑である．本研究では，楕円を含む二次曲線の方程式を解く方法を用いる．

以下がその流れである．また，以下のアルゴリズムは制約条件を加えるまたは，変えることで一般の他の二次曲線の場合にも当てはまる．

楕円を含む一般的な二次曲線の方程式は

$$Ax^2+2Bxy+Cy^2+2Dx+2Ey+F=0 \quad (3)$$

で与えられる[4]．しかし，ここでは，$C=1$としても

一般性は失われないため，上の式の両辺を$C=1$で割り，改めてパラメータを設定すると，以下のように変形することができる．

$$y^2 = -Ax^2 - 2Bxy - 2Dx - 2Ey - F \quad (4)$$

ここで楕円あてはめ問題を定式化すると，

「エッジ点列$P=\{p_1,p_2,p_3,,,p_n\}$が与えられた時，各々の点$p_i=(x_i,y_i)$が、式(4)を満たすようなパラメータA,B,D,E,Fの組を求めればよい.」ということになる．

ここでは，以上の問題を解くためにいくつかの定義を行う．入力データベクトル$X_i=(-x^2,-2xy,-2x,-2y,-1)^T$，パラメータベクトル$\beta=(A,B,D,E,F)^T$，出力$Y_i=y_i^2$と定義すると，上の式(4)は$Y_i=X_i^T\beta$と書くことができる．ここで評価関数

$$E=\sum_{i=1}^{N}(Y_i-\hat{y}_i)^2=\sum_{i=1}^{N}(Y_i-X_i^T\hat{\beta})^2 \quad (5)$$

を導入する．$\hat{y}$は推定されたパラメータベクトル$\hat{\beta}$を元に推定された出力である．このEを最小にする$\hat{\beta}$が求めるパラメータベクトルとなる．さらにデータ行列，出力ベクトルを次の様に定義する．

$$X=\begin{pmatrix}X_1^T & X_2^T & \cdots & X_N^T\end{pmatrix}^T$$

$$Y=\begin{pmatrix}y_1^2 & y_2^2 & \cdots & y_N^2\end{pmatrix}^T$$

評価関数Eを改めて書き直すと次のようになる．

$$E=(Y-X\beta)^T(Y-X\beta) \quad (6)$$

両辺をβで微分すると、求めるパラメータベクトルβは

$$\frac{\partial E}{\partial \beta}=2X^T(Y-X\beta)=0 \quad (7)$$

を満たすものである．

これより，推定されたパラメータベクトルは

$$\hat{\beta}=(X^TX)^{-1}X^TY \quad (8)$$

である．

パラメータベクトルβが求まれば，次に楕円の中心座標，傾き，長軸，短軸の長さなど一般に知られる楕円のパラメータを求めることができる．

二次曲線を与える方程式は式(3)より

$$Ax^2+2Bxy+Cy^2+2Dx+2Ey+F=0 \quad (3)$$

である．そこで，以下のようにベクトルr,cと行列Sを

$$r=\begin{pmatrix}x\\y\end{pmatrix},S=\begin{pmatrix}A & B\\B & C=1\end{pmatrix},c=\begin{pmatrix}D\\E\end{pmatrix} \quad (9)$$

と定義すると，二次曲線の方程式は

$$(r,Sr)+2(c,r)+F=0 \quad (10)$$

と書くことができる．ここで，Sは対象行列であるので，固有値λ_1,λ_2，固有ベクトルu_1,u_2を持つ，このベクトルu_1,u_2は二次曲線の主軸方向を与えることがわかっている．つまりこの主軸方向が楕円の傾きθになる．主軸方向が求まったら，固有ベクトルu_1,u_2が座標軸方向に一致するように全平面を原点の周りに回転し，平行移動させ適当に座標軸を取り直すと，新たな原点を中心とする楕円の方程式

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1 \quad (11)$$

が得られる．この過程で，楕円の中心(x',y')，長軸，短軸の長さの半分a，bが求まる．

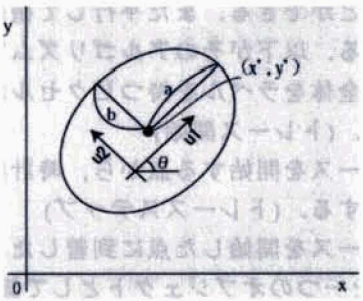


図6 楕円パラメータの定義

楕円あてはめアルゴリズム

1．nが閉じた領域のエッジ点の総数Nになるまで以下2-5を繰り返す

2．エッジ点列$P=\{p_1,p_2,p_3,,,p_n\}$(n=トレースが終わった点の数)より，データ行列，出力ベクトルを定義する

$$X=\begin{pmatrix}X_1^T & X_2^T & \cdots & X_n^T\end{pmatrix}^T$$

$$Y=\begin{pmatrix}y_1^2 & y_2^2 & \cdots & y_n^2\end{pmatrix}^T$$

3．以下の正規方程式(8)を解く

$$\hat{\beta}=(X^TX)^{-1}X^Ty$$

4．求まったパラメータベクトル$\hat{\beta}$を投票空間に投票する．

5．次の点をトレースする．

6．投票空間で最も投票を得たパラメータを楕円のパラメータとする．

以上が楕円あてはめアルゴリズムであるが，検証実験では，計算量を減らすため，すべてのトレース点

で正規方程式(8)を解くのではなく，トレース点を間引いて解を求めた．

以下の図７が楕円を検出した例である．

図７　楕円検出

左：入力画像　中左：ラベル付け
中右：楕円当てはめ右：検出楕円

4. 実験

4.1.1. 静止画像

まず，円検出，楕円検出の有効性を確かめるために静止画に対して検証実験を行った．

実験に用いた画像は，2001 年電気情報通信学会 PRMU アルゴリズムコンテスト「道路交通標識の認識」[3]で提供された画像を用いた．対象とする画像は 800x600 ピクセル，24 ビットカラー画像 118 枚である．これらの静止画に対してアルゴリズムを適用し確認した．

4.1.2. リアルタイム動画像

次に同じアルゴリズムを動画に対して適用した．実験はオンライン，オフラインの両方で行った．オフライン実験は，事前にデジタルビデオカメラで撮影した動画像をテープに保存しておき，PC 上ですべてのフレームに対して本アルゴリズムを適用した．オンラインでは自動車にコンピュータとデジタルビデオカメラを持ち込み走行中にアルゴリズムを適用した．どちらの実験でも入力画像サイズは 320x240 である．

5. 結果

5.1. 静止画像

以下に静止画像に対する実行結果を示す．入力画像より，道路交通標識部分のみを残しそれ以外を黒くした．このように入力画像から道路標識部分だけが切り出されていることがわかる．円形切り出し(図９)と楕円切り出し(図１０)で若干の差が見られる．図１０の様に楕円切り出しの方が正しく切り出されている．計算時間は，画像中に含まれる円，楕円の数によって変化するが，Pentium4 プロセッサ 2.8Ghz の CPU を用いて，どちらも 200~250ms である．楕円あてはめ手法はアルゴリズム中に間引きを行い，同様な計算時間を実現させたため手法による大きな差は無い．本アルゴリズムの有効性を確かめるためにシステムの精度を評価した．道路交通標識の切り出し率を以下のように定義する．

$$\text{切り出し率}=\frac{\text{切り出し数}}{\text{目視による道路高越標識の数}}$$

この切り出し率を円，楕円の切り出しそれぞれについて求める．以下の表２がその結果である．切り出しの際に道路交通標識以外の領域もまれに含まれるが，これらについてはカウントに入れなかった．なぜなら，切り出さなければいけない標識がきりだされないよりも，標識でない物が切り出される方がドライバーに提供する情報の選択肢増やすことができると考えたからである．この表より，楕円あてはめの方が精度がよいことがわかる．リアルタイム動画像に対する実験ではこれを元に，楕円あてはめ手法による検出を行った．

表２　切り出し率

	すべての標識の数	切り出した標識の数	切り出し率
赤の円検出＋青，黄標識の切り出し	208	165	79.3%
赤の楕円検出＋青，黄標識の切り出し	208	171	82.2%

図８　入力画像

図９　円検出

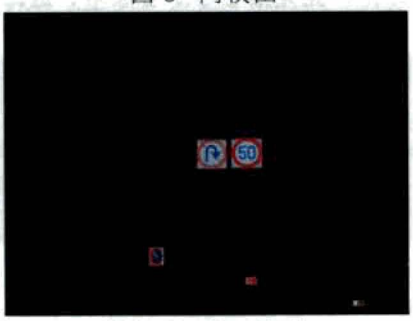


図１０　楕円検出

検出できなかった入力画像を以下に示す．これらの画像で検出できなかった原因には，背景と標識とが混

ざってしまい色のラベル付けの時点で周囲と同じラベルが付けられてしまったことや，逆光によりほとんど黒くしか見えないことが原因であると考えられる．また，色褪せたもの，汚れている物も切り出せなかった．

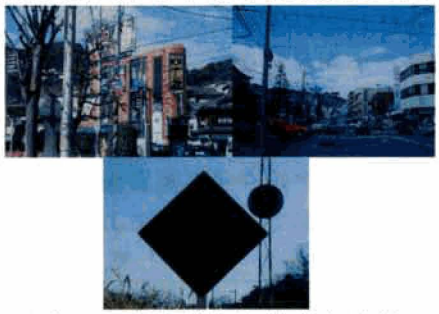


図１１　切り出しに失敗した例

5.2. リアルタイム動画像

動画像に対しても同様のアルゴリズムを適用する．オフライン実験では先も述べたように，すべてのフレームに対して，オンライン実験では，10fps で次々と画像を処理していった．動画像処理で用いる入力画像サイズは 320x240 ピクセルである．この大きさの画像１枚を処理する時間は，静止画像を処理した PC と同等の PC で 30~50ms であり，リアルタイムで処理が可能である．以下の図１２は走行中に検出した道路交通標識の例である．

図１２　動画像処理過程

5.3. リアルタイム動画像-夜間-

5.2 節と同じ実験を夜間に撮影された動画像に対してオフライン実験をおこなった．撮影時間以外の実験環境は同じである．以下の図１３はその夜間シーンに対して，HSB によるラベル付けを行った結果である．このように，一見見落としがちな暗い道路交通標識でも，色のラベル付けで浮き出させることができる．

図１３　夜間動画処理過程

6. まとめ

以上の実験より，基本原理に色情報を用いることで，高速で高い精度を持ちかつ取りこぼしの少ない道路交通標識切り出しアルゴリズムを示すことができた．今後の展開としては，円形，楕円形以外の道路標識の識別，切り出しと，道路標識の認識に移っていきたいと思う．また，本稿の動画像処理は，フレームごとの相関を全く無視した静止画像に対するアルゴリズムをそのまま動画像に適用した物である．動画像ではフレーム間の相関が一般にあるので，それを考慮に入れた動画像処理を行っていきたい．具体的には，以前のフレームで認識された情報を元に次のフレームでの検索範囲を狭めるといった手法を考えている．また，切り出された道路交通標識の認識も行っていきたいと考えている．

参考文献

[1] 国土交通省 ITS ホームページ
http://www.mlit.go.jp/road/ITS/j-html/

[2] Advanced Cruise-Assist Highway System Research Association http://www.ahsra.or.jp/

[3] "2001 年 PRMU アルゴリズムコンテスト「道路交通標識の認識」実施報告とその入賞アルゴリズムの紹介"信学技報，Vol.101 No.525 pp.125-132,2001 年 12 月

[4] 金谷健一，空間データの数理-３次元コンピューティングに向けて-，朝倉書店，東京，1995.

[5] 松浦大祐，山内仁，高橋浩光，"特定色判別と領域限定を用いた円形道路標識の認識"，信学論 D-II，Vol.J85-D-II,No.6 pp.1075-1083，2002 年 6 月

[6] 井上泰夫，石川直人，中島直人，"道路標識の自動認識"，電子情報通信学会技術研究報告，Vol.101，No.627，pp.67-72，2002 年 1 月

社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

PRMU-2003-258, pp.13-18, 2004-03

信学技報
TECHNICAL REPORT OF IEICE

Opponent-Colorフィルタを用いた道路交通標識認識システム

石塚 裕† 平井 有三‡

†筑波大学大学院理工学研究科 〒305-8573 つくば市天王台 1-1-1 筑波大学理工学研究科
‡筑波大学電子情報工学系 〒305-8597 つくば市天王台 1-1-1 筑波大学電子情報工学系

E-mail: †yishizuka@viplab.is.tsukuba.ac.jp ‡hirai@is.tsukuba.ac.jp

あらまし 自動車に搭載されたカメラ映像から，道路標識を実時間で切り出し，認識するアルゴリズムを提案する．道路標識の切り出しでは，道路標識の色情報を用いることにより，効率よく高速に行えることを示す．道路交通標識の識別を行うために，標識内の記号部分を抽出する必要がある．そのために，ヒトの眼の色受容野をモデル化した反対色フィルタを提案し，解像度の低い画像でも記号部分の特徴抽出が行えることを示す．認識では，記号部分の構造的特徴を抽出し，決定木により認識するアルゴリズムを提案し，高い精度で認識できることを示した．

キーワード ITS, 道路交通標識, 認識, 実時間, ハフ変換, 構造的特徴抽出

Recognition System of Road Traffic Signs using Opponent-Color Filter

Yutaka ISHIZUKA† and Yuzo HIRAI‡

†Master Program of Science and Engineering, University of Tsukuba 1-1-1
Tennoudai,Tsukuba-shi,Ibaragi,305-8573 Japan
‡Institute of Information Sciences and Electronics, University of Tsukuba 1-1-1
Tennoudai,Tsukuba-shi,Ibaragi,305-8597 Japan
E-mail: †yishizuka@viplab.is.tsukuba.ac.jp, ‡hirai@is.tsukuba.ac.jp

Abstract A novel algorithm for the segmentation of symbol parts in road signs is proposed. After detecting a read annular object using standard HSB color coordinates, a biologically inspired opponent-color filter is applied to the detected region. The opponent-color filter used in this work consists of a small excitatory red region surrounded by an annular inhibitory green region. By applying discriminant analysis to the filter output, the red annular frame of a road sign can be segmented. By applying the discriminant analysis to the interior part of annulus again, symbol parts of road signs can be segmented effectively. Road signs are identified by a standard decision tree and verified that opponent-color filter can lead accurate recognition even for small and degraded road signs in video images.

Keyword ITS, Road Traffic Signs , Recognition, Real-time, Hough transform, Structural-feature extraction

1. はじめに

日本では，年間約 80 万件の交通事故が起こっており，そのうち 1 万人もの人が死亡し，100 万人もの人が負傷している(平成 7 年)．それらの事故の原因の半数以上が，「危険な事象の発見の遅れ」や「ドライバーの判断の誤り」である．本研究の目的は，事故の大半を占める「回避可能であったはずの事故」を減らす交通支援システムを作ることである．

交通支援システムを実現する手段には，電磁波，レーダー波などから得られる情報をコンピュータを用いて認識するシステム等も考えられるが，本研究では，人が普段行っているような，眼から入力される情報を処理するアプローチを用いる．その理由は，設備が最小限で済むということである．電磁波やレーダー波を用いる手法は，道路と自動車の両方に設備が必要であり，設備がない場所では機能しない．しかし，本アプローチは自動車への設備のみでどのような環境にでも適応可能である．本研究では車載カメラから得られる映像を処理し認識するシステムを提案，評価する．

2. システム構成

ドライバーに，より安全で正確な情報を与えるために必要な技術は，1．高速に処理することができること．2．高い精度で認識することができること．3．情報に取りこぼしが無いこと．4．周囲の環境の変化にロバストであること．が要求される．特に高速でかつ高い精度で認識ができることは，交通の安全上絶対不可欠な要素である．

2.1. 対象標識

本研究で対象とする標識は速度規制標識(30/km/h, 40km/h, 50km/h)，駐車禁止，駐停車禁止，追い越し禁

止，通行止め，車両通行止め，U ターン禁止の 9 種類である．これらの標識は，日常よく目にする標識である．

2.2. システム概観

本システムの構成を図 1 に示した．車載ビデオカメラで画像を取り込み，道路標識部分をシーンより切り出し認識し，最後にドライバーへ情報提供を行う．

走行中に自動車に搭載されたカメラから，画像を取り込み，次節以降で述べる動画像処理を行い，画像中より道路標識候補を切り出す．そして，それぞれ切り出された「標識が何であるか？」を認識させる．このようにして，低レベル画像情報から，「速度制限」，「駐車禁止」などの高度レベルな情報へと変換して行く．高レベル情報は，ドライバーの運転を支援するために利用する．

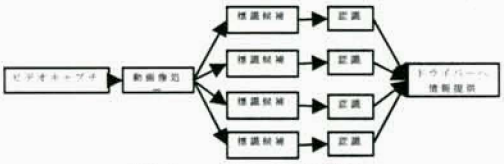


図 1　システム概要

3. 動画像処理

道路標識を認識するためには，カメラ画像から標識部分を切り出さずに認識する手法も先行研究にあるが[11]効率はよくない．なぜならば，画像全体に占める標識の領域が狭いため，標識以外の部分も処理することになる．事前にその領域が標識でないと分かれば，画像に含まれる標識の認識率はよくなるであろう．そのような理由から，画像から標識部分を切り出す必要がある．本節ではそのための動画像処理について述べる．

3.1. 色相によるラベル付け

道路交通標識は，「色」で背景から容易に区別できるように設計されている．そこで，本研究では，標識に使用されていない色に着目し，画像から標識部分だけを残す処理をする．また，一般にカメラや画像処理で用いられる RGB 表色は．色相を一つのパラメータで表すことができない．そこで，色相，彩度，明度のパラメータである HSB 表色系を用いて，画像の各ピクセルにラベル付けを行う．ここでは，本研究で対象としている標識以外にも対応できるようにラベル付けという言葉を用いるが，処理としては色相ごとの二値化と同等である．また，ラベルには RGB 値を用い，後段の処理を容易にした．

表 1　ラベル付け対応表

ラベル	色相(0~1.0)	彩度(0~1.0)
赤(255,0,0)	0~0.08 0.9~1.0	0.2 以上
黄(255,0,255)	0.08~0.2	0.5 以上
緑(0,0,255)	0.2~0.5	0.7 以上
青(0,255,0)	0.5~0.75	0.5 以上
無し(0,0,0)	上記以外	上記以外

(明度は 0.1 以上 1.0 以下とした)

3.2. 標識部分の切り出し

本節では，色のラベル付け(赤のみ)処理と，ノイズ除去後の画像から道路交通標識であると思われる領域を切り出す方法を述べる．標識領域の切り出しには，ラベルのエッジをトレースする方法を用いた．エッジのトレースは，図形の縁の特徴を抽出するため，円以外の図形にも適用可能である．本研究では，先にも述べたように対象としている標識は円形の道路交通標識であるので，トレースされたエッジを用いた円のあてはめを行った．

3.3. 円のあてはめ

エッジのトレースにより領域の切り出しが可能であるが，円形な道路交通標識を切り出すためには，それが円かどうか調べる必要がある．ここでは，エッジをトレースしている間，その領域が円であるかどうかを調べる Hough 変換に基づくアルゴリズムを用いた．

原点を通る円の方程式は，以下のように与えられる．

$$x^2 + y^2 - 2gx - 2fy = 0$$

(g, f):円の中心座標

原点とその他 2 点を通る円は，

$$\begin{cases} x_1^2 + y_2^2 - 2gx_1 - 2fy_1 = 0 \\ x_2^2 + y_2^2 - 2gx_2 - 2fy_2 = 0 \end{cases}$$

である．

この連立一次方程式を解くことで，円の中心 を求めることができる．また，円の半径は，円が原点を通るという制約のため，半径は $r = \sqrt{g^2 + f^2}$ である．(図 2)

トレースを開始する点を原点とする新たな座標系を設定し，トレース中にエッジ点列より 2 点をランダムに選び出し．この方程式を解く．求まった (g, f) が円のパラメータとなり，これを投票空間に投票しもっともらしいパラメータを選ぶ．

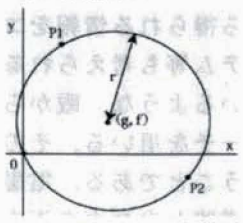


図 2　円のパラメータ算出方法

このようにして円のパラメータを求めることができるが，円でないものも無理矢理あてはめることができる．そこであてはめ誤差について検証した．

円のあてはめ誤差を次の様に定義する．(図 3)

$$E=\frac{1}{(\text{エッジ点の総数})}\sum_{\text{すべてのエッジ}}(1-\frac{r}{q})^2$$

図 3　円の誤差計算方法

この誤差 E が小さいほどよく当てはまっているということになる．この値がある閾値以下の場合円と判断することにする．

4. 標識内部のセグメンテーション

本研究では認識に図形の構造的特徴を用いる．よって標識内部を記号部分と背景にセグメンテーションする必要がある．しかし，これまでに述べた二値化法では，道路交通標識の外側の赤い部分を切り出すための処理であり，標識内部をセグメンテーションすることができない．そこで，内部をセグメンテーションするための，人の視覚系に学んだ反対色フィルタを開発した．

4.1. Opponent-Color Filter

Opponent-Color Filter とは，ヒトの眼の受容野をモデル化したフィルタである．この受容野は，中心に赤い光が当たると発火し，同時に周辺に緑の光が当たると抑制される細胞である．

ヒトの脳にある色を知覚する細胞は，三つの錐体からなっている．それらは L 錐体，M 錐体，S 錐体であり，簡単に言うと，光の三原色である RGB にそれぞれ反応する．典型的な受容野は，興奮性，抑制性の領域が同心円状に広がっている．L 錐体から興奮性の反応を受け取る中心部分と，M 錐体から抑制性の反応を受け取る周辺部分を持つ受容野を L-M Color Opponent 細胞と呼ぶ．このような構造を持つ細胞は，大きな面積をもつ白や，黄色，青の領域には反応しない．道路交通標識では，これらの色は標識内部の背景を構成する色である．よって Opponent-Color Filter は，標識内部を記号部分と背景に分けるフィルタになる．

Opponent-Color 細胞を再現するフィルタは，符号を逆にしたラプラシアンガウシアンフィルタであり，以下式で表される．[10]

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}\quad(\text{ガウシアンフィルタ})$$

$$OpponentColorFilter(x,y)=-\nabla^2G(x,y)=\frac{1}{\pi\sigma^4}\left[1-\frac{x^2+y^2}{2\sigma^2}\right]$$

模式的に表すと次のようになる．(図 4)

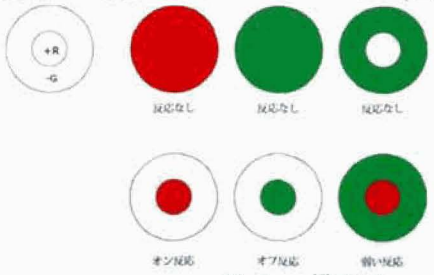


図 4　模式図

このフィルタの中心に，赤や白といった光があたると出力値は大きくなり，逆に青い光が当たると出力は押さえられる．このようにして，切り出された道路標識より，赤い部分を強調し，青成分を抑制することで，その中身を切り出すことができる．次に実際の画像でこのフィルタリング処理をした結果を示す．(図 5)

図 5　Opponent-Color フィルタをかけた画像
(左：切り出した画像．右：フィルタ後の画像)

4.2. セグメンテーションプロセス

続いてこのフィルタリングされた画像をセグメンテーションしなければならない．ここでは，Opponent-Color Filter と判別分析二値化法を用いた二値化処理の流れについて述べる．その手順は以下の通りである．

1．切り出した領域に，Opponent-Color Filter を施す．

2．同じ領域に判別分析二値化法を施す．この際，図の部分の中心領域をマスクとして残しておく．

3．1 でつくった画像を，2 で作ったマスク領域の部分にだけ判別分析二値化法を施す．

このような手順で，標識全体の画像から，標識内部を鮮明にすることができる．以下の図 6 がその手順を図示したものである．

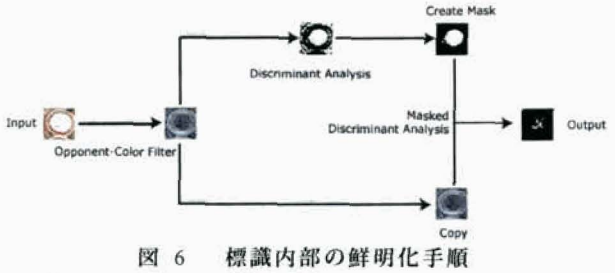


図 6　標識内部の鮮明化手順

5. セグメンテーション結果の比較

ここでは以上の手順で二値化を行った結果と，その他の手法の結果を比較する．比較のための手法として，

Mean Shift Filtering[8] と Bilateral Filtering[9]を用いた．これらの手法はどちらも，エッジを保つ平滑化フィルタの一種であり，領域分割手法でもある．以下の表 2 がそれらの比較である．また，切り出した標識の色相と明度による判別分析二値化法も比較を行った．

表 2　各手法の比較

解像度	入力画像	Mean Shift	Bilateral	色相による判別分析	明度による判別分析	本手法
低						
高						

このように本手法の結果は良好なものであると言える．従来手法では，ある程度の領域分割は可能であるが，本来必要な標識内部の鮮明化がうまくいかない．また，色相による判別分析二値化法は標識内部を鮮明にしているようにみえるが，ノイズがのってしまう．明度による判別分析二値化法は標識内部と外部を区別しているだけである．しかし，Opponent-Color Filter と判別分析二値化法を組み合わせた本手法は，標識内部のみをきれいに取り出すことができることがわかる．

以下図 7 左に示したヒストグラムは，図 7 中央の画像の明度ヒストグラムである．

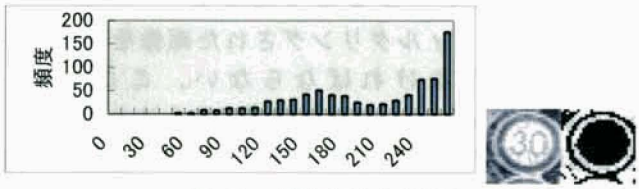


図 7　明度ヒストグラム

グラフには双峰性を見ることができるが，この画像に判別分析二値化法を適用しても，右図に示したように標識内部を二値化できない．

円のあてはめから，標識の内部と推測される領域を切り出すことは可能なので，標識内部の明度ヒストグラムと判別分析二値化法による二値化結果を見てみる．(図 8)

図 8　標識内部の明度ヒストグラム

このように，セグメンテーションできない．これは，完全に標識の内側と外側を分けられなかったことが原因であると言える．

続いて，本研究で提案する Opponent-Color Filter によりフィルタリングした画像のヒストグラムを見てみる．（図 9）

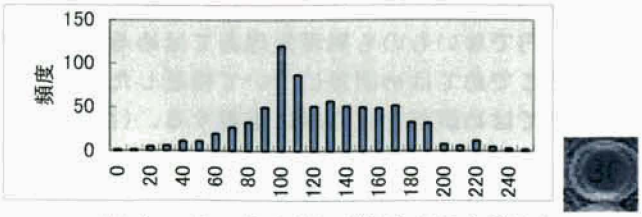


図 9　フィルタリング後のヒストグラム

図 9 では双峰性を見ることができる．ここで判別分析二値化法を行うとその閾値は 150-160 の間に存在し，この閾値を元に二値化すると，次の画像が得られる．(図 10)

図 10　Opponent Filter 後の判別分析法

この画像から，中心部分を残すマスクを作成する．作成したマスク部分のフィルタ後の出力値のヒストグラムを見てみると以下の図 11 の様になる．

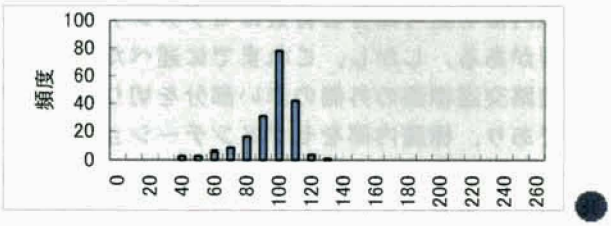


図 11 Opponent-Color Filter 後の標識内部のヒストグラム

このように単峰性であるが，判別分析二値化法によりフィルタの出力値が低い図と高い地の部分に分けることができた（図 12）．

図 12　セグメンテーション結果

6. 認識

標識内部の記号部分を抽出した後，どの標識かを認識する．認識は，記号部分の構造的特徴を用いた決定木によって行う．構造的特徴を用いる方法は，標識の大きさや，回転，二値化の程度に対して頑健であり，テンプレートマッチング等の手法に比べても高速に処理ができる．

6.1. 駐車禁止・駐停車禁止・通行止め・車両通行止め・U ターン禁止

駐車禁止・駐停車禁止・通行止め・車両通行止め・U ターン禁止の標識については下の図 13 の様に画像の特定の部分を走査して特徴を抽出した．画像を矢印の方向に走査し，ピクセルの有無で符号化する．この抽出された特徴は，ここで対象としている標識について一意に定まり，標識の大きさ，二値化の程度に対してロバストである．

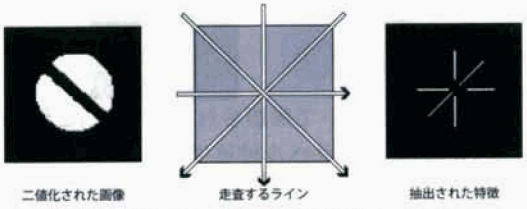


図 13　特徴抽出方法 1

図 13 にように走査することで，同じピクセルの連続を符号化する．上の図 13 左のような駐車禁止の標識の場合は，次のように符号化される．

駐車禁止・車両通行止め

横のライン：黒白黒白黒

縦のライン：黒白黒白黒

右上から左下：黒白黒白黒

左上から右下：黒

入力も同様に符号化すると，入力とこれらの照合は，単なる文字列の場合のパターンマッチングと同じ枠組みとなり，高速に処理ができる．

6.2. 追い越し禁止

追い越し禁止については，“画像上で二つの領域に分割され，各領域の面積が大きく異なる”という特徴を用いた．二値画像を X 軸方向に射影したヒストグラムを作り，ヒストグラムの谷を見つける(図 14)．この谷が追い越し禁止の標識を左右に分ける点になる．この点の左右での矢印部分の面積を比較して認識した．

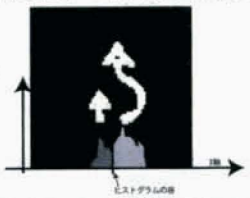


図 14　特徴抽出方法 2

6.3. 速度規制(30km/h,40km/h,50km/h)

速度規制標識については，Glucksman による背景特徴の符号化と位相構造化特徴を組み合わせ簡略化したものを用いた．たとえば，下の図 15 で太い矢印上を走査したとき，それぞれの点の左右に図が存在するか，地が存在するかを探す．もし，右側には図しかなく，左側には図しかない場合，この文字は，“3”と判別される．

図 15　特徴抽出方法 3

6.4. 決定木

以上のルールを決定木を用いて分類する．決定木は以下の図 16 の通りである．本研究で対象としている標識がまだ少ないが，今後同様にして増やしていくことが可能である．

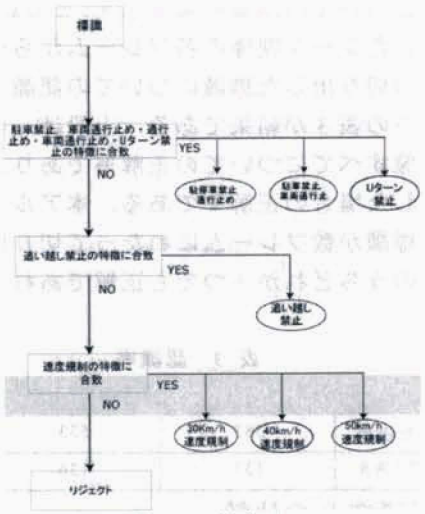


図 16　決定木

7. 実験

以上のアルゴリズムを実装して実験を行った．実験には，ビデオカメラをのせた三脚を自動車の助手席に固定して撮影し，実験室の PC によって処理を行った．デジタルビデオカメラは，SONY(DCR-TRV900 NTSC)を用い，使用した PC は，Dell Precision 340，Intel Pentium4 2.8Ghz,2GB Memory である．IEEE1394 端子にて PC に映像を取り込む．キャプチャする画像のサイズは，720x480 ピクセルのフルカラー画像である．この取り込んだ画像に対して順次アルゴリズムを適用してゆく．カメラはズーム機能は使用しなかった(f=4.3mm)．この画像から切り出される標識のサイズは，20-60 ピクセルの大きさである．

7.1. シーンからの道路標識部分の切り出し率

道路交通標識を認識するために，映像から道路交通標識候補を画像より切り出す．その性能を評価するために切り出し率を用いて性能の評価を行う．切り出し率とは映像に含まれる標識の数に対する，本研究のラベル付けと円あてはめアルゴリズムにより映像より切り出された標識の割合である．

市街地を約 30 分走行して撮影された映像から，切り出された標識の割合は，76.11％であった．

7.2. 実験結果-切り出しから認識までの時間-

ここでは，本研究の課題であるリアルタイム性についての結果について述べる．以下にその結果を示す．

- 標識 3 個を切り出す時間：約 120ms
- セグメンテーション：5ms 以下
- 認識：5ms 以下

以上より，約 150ms/frame で処理することができ，実時間で認識処理が可能である．

7.3. 認識結果

最後に，切り出しおよびセグメンテーションが終了した画像についての認識結果を示す．この認識結果は，実験に用いたシーン映像の各フレームから標識を切り出し，その切り出した標識についての認識率となっている．以下の表 3 が結果である．上段は，切り出された標識画像すべてについての正解率であり，下段は標識に着目した場合の正解率である．本アルゴリズムでは，同じ標識が数フレームにわたって切り出される．その画像のうちどれか一つでも正解であれば正解と見なした．

表 3　認識率

	すべての標識	正解数	認識率
切り出された画像	583	533	91%
存在する標識での割合	137	136	99%

7.4. 先行研究との比較

以上の結果をふまえて先行研究[6]との比較を行った．

この研究では，本研究と同様に車載カメラにより映像を撮影し，リアルタイムで認識している．また，本研究と同様に HSV 表色系という HSB 表色系と同様の表色系を用いている．処理の流れは，本研究と同様に，画像からの標識領域を切り出し，認識というプロセスである．認識には切り出した画像のエッジを用いたテンプレートマッチングを行っている．結果の方は，正検出率(本研究での切り出し率)が 97.5%，正認識率(切り出した画像の認識率)が 96%である．認識の処理速度は，1 フレームあたり 120ms～150ms であり，リアルタイム処理が可能である．ただし，処理しているシーンが異なるので，認識率などの直接的な比較はできない．

8. 結果の考察と今後の課題

8.1. 結果の考察

切り出した標識の認識率は，標識そのものについては 99%，切り出した画像 1 枚 1 枚については 91%と概ね良好な結果を出しているが，切り出し率がそれとくらべて低い．切り出し率が高くないため標識の取りこぼしが発生し，ドライバーに必要な情報を提供できなくなってしまう．このように切り出しがうまくいかない原因がいくつかあることがわかった．その一つに，逆光がある．逆光のシーンでは，人間の眼には赤く見える標識でも，ビデオカメラを通して撮影された画像は黒く見えてしまうことが分かった．このような画像に色相によるラベル付けを行うと一つの標識が二つの領域に別れてしまうことが分かる．（図 17）

図 17　逆光のシーンで撮影された画像(左)と色相によるラベル付け(右)

このような状況では，円のあてはめができず，切り出すことができない．これを改善するためには，円のあてはめで局所的な空間で行っていた投票を，画像全体のグローバルな空間に投票することで，色相によるラベル付けで分断されてしまった標識を一つの標識として切り出すことができるのではないかと考えている．

8.2. 今後の課題

今後の課題は，上で述べたこと以外に，対象とする標識を増やすことがあげられる．現在は 9 種類で行っているが，これでは少ない．実用的には，さらに多くの標識を対象とする必要がある．また，現段階では，ドライバーへの情報提供インターフェイスにはふれていない．認識結果をドライバーへ提供するにはどのようなインターフェイスが良いか，今後考察する必要がある．

文　献

[1] 国土交通省 ITS ホームページ
http://www.mlit.go.jp/road/ITS/j-html/

[2] Advanced Cruise-Assist Highway System Research Association
http://www.ahsra.or.jp/

[3] ”2001 年 PRMU アルゴリズムコンテスト「道路交通標識の認識」実施報告とその入賞アルゴリズムの紹介” 信学技報, Vol.101 No.525 pp.125-132,2001 年 12 月

[4] 金谷健一, 空間データの数理-３次元コンピューティングに向けて-, 朝倉書店, 東京, 1995.

[5] 松浦大祐, 山内仁, 高橋浩光,” 特定色判別と領域限定を用いた円形道路標識の認識”, 信学論 D-II, Vol.J85-D-II,No.6 pp.1075-1083, 2002 年 6 月

[6] 井上泰夫, 石川直人, 中島直人,” 道路標識の自動認識”, 電子情報通信学会技術研究報告, Vol.101, No.627, pp.67-72, 2002 年 1 月

[7] 大田雄也,” HSB 表色系を用いた道路交通標識切り出し処理の SVM による学習”, 平成 14 年度筑波大学第三学群情報学類卒業論文

[8] D.Comaniciu and P.Meer. “Mean shift : A robust approach toward feature space analysis”, IEEE Trans. On Pattern Analysis and Machine Intelligence,PAMI-14(5):603-619,2002

[9] C.Tomasi and R.Manduchi. “Bilateral filtering for gray and color images.”, In ICCV,pages 839-846,1998

[10]平井有三,” 視覚と記憶の情報処理”, 倍風館, 東京, 1995

[11]内村圭一, 木村英雄, 脇山慎也. “道路情景カラー画像における円形道路標識の抽出および認識”, 信学論, Vol.J81-A,No.4 pp.546-553,1998 年 4 月

社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

ITS2004-48, IE2004-182, pp.45-50, 2005-02

信学技報
TECHNICAL REPORT OF IEICE

車載カメラ画像による道路交通標識のリアルタイム認識

馬場 今日子† 平井 有三‡

†筑波大学第三学群情報学類 〒305-8573 つくば市天王台 1-1-1 筑波大学第三学群情報学類
‡筑波大学システム情報工学研究科 〒305-8573 つくば市天王台 1-1-1 筑波大学システム情報工学研究科
E-mail: †baba@viplab.is.tsukuba.ac.jp, ‡hirai@cs.tsukuba.ac.jp

あらまし 車載カメラ画像から赤い円形の道路交通標識を切り出し，リアルタイムで認識するアルゴリズムを提案する．HSB 表色系を用いて標識を切り出した後，反対色フィルタを標識内部を鮮明化するためにかける．そこに判別分析二値化法を二度用いることによって，標識内部がセグメンテーションされる．そして，セグメンテーションされた標識に対して構造的特徴を用いた決定木を使って認識を行う．その結果，約 96%の認識率で認識できた．

キーワード ITS, 道路交通標識，認識，リアルタイム，構造的特徴抽出

Recognition System of Road Traffic Signs by a Color Video Camera

Kyoko BABA† and Yuzo HIRAI‡

† College of Information Sciences, University of Tsukuba 1-1-1 Tennoudai, Tsukuba-shi, Ibaragi, 305-8573 Japan
‡ Institute of Infommation Sciencesk and Electronics, University of Tsukuba
1-1-1 Tennoudai, Tsukuba-shi, Ibaraki, 305-8573 Japan
E-mail: †baba@viplab.is.tsukuba.ac.jp, ‡hirai@cs.tsukuba.ac.jp

Abstract A novel algorithm for the recognition of Japanese road sign symbols with red annular boundaries is proposed. After detecting a red annular object using HSB color coordinates, a biologically inspired opponent-color filter is applied to extract the symbol parts of road sign. By applying discriminant analysis to the filter output, the inside region circumscribed by red annular boundary can be extracted. By applying the discriminant analysis again to the inside region, the symbol part of road sign can be segmented. The segmented symbol is classified by a decision tree and recognition rate of about 96% achieved.

Keyword ITS, Road Traffic Signs, Recognition, Real-time, Structural-feature extraction

1. はじめに

日本では，年間 94 万件もの交通事故が起こっており，そのうち 7 千人が死亡，118 万人もの人が負傷している(平成 15 年)．その主な事故原因は，危険な事象の発見の遅れやドライバーの判断の誤りである．本研究の目的は，ドライバーの認知，判断をサポートする交通支援システムを作ることである．

交通支援システムを実現する手段には，電磁波，レーダー波などから得られる情報を用いて認識するシステムも考えられるが[1]，本研究では人が普段見ているような視覚情報を処理するアプローチを用いる．なぜならば，設備が最小限ですむということである．電磁波やレーダー波を用いる手法は道路と自動車の両方に設備が必要であり，設備のない場所では機能しない．しかし，本研究では自動車への設備のみでどのような環境にでも適応可能である．よって，車載カメラから得られる画像を処理し，認識するシステムを提案，評価する．

2. システム構成

ドライバーにより安全で正確な情報を与えるために必要不可欠な要素は，1) 高速に処理ができること，2) 高い精度で認識すること，3) 情報に取りこぼしがないこと，4) 周囲の環境の変化にロバストであることである．特に高速かつ高い精度で認識できることは，ドライバーが実際運転している際，絶対不可欠な要素である．

2.1. 対象標識

本研究で対象とする標識は，速度規制標識(30km/h,40km/h,50km/h)，駐車禁止，駐停車禁止，追い越し禁止，U ターン禁止の 7 種類である．これらの標識は日常よく目にする標識である．

2.2. システム概要

本システムの概要を図1に示した．車載ビデオカメラで画像を取り込み，道路標識部分をシーンより切り出し，認識しドライバーへ情報提供を行う．

走行中に自動車に搭載されたカメラから，画像を取り込み次節以降で述べる動画像処理を行い，画像中より標識候補を切り出す．そして，それぞれ切り出された標識が何であるかを認識し，標識であると認識されたものをドライバーに情報提供する．

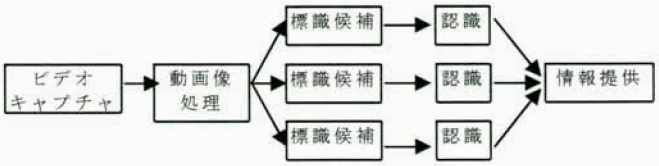


図1　システム概要

3. 動画像処理

道路標識を認識するため，まずカメラ画像から標識部分を切り出す．これは，画像全体に占める標識の領域が狭いので，標識以外の広い部分を処理するのは非効率的であるからである．

3.1. 色相によるラベル付け

道路交通標識は，その特徴的な色で背景から容易に区別できるように設計されている．そこで，本研究では，標識に使用されている色に着目し，画像から標識部分だけを残す処理をする．

一般にカメラや画像処理で用いられる RGB 表色系は，色合いを一つのパラメータで表すことができない．そこで，色相，彩度，明度のパラメータである HSB 表色系を用いて，画像の各ピクセルにラベル付けを行う．こうすることにより，明るさに対してもロバストになる．本研究では，赤色の円形の標識のみを対象としているため，赤色のラベル付けを行う．色相が 0～0.08 もしくは 0.9～1.0，彩度が 0.2 以上，明度が 0.1 以上のとき赤色とした．

3.2. 標識部分の切り出し

本節では，色のラベル付け処理をし，ノイズを除去した画像から道路交通標識であると思われる領域を切り出す方法を述べる．標識領域の切り出しには，赤色にラベル付けされた部分のエッジをトレースする方法を用いた．エッジのトレースは，図形の縁の特徴を抽出するため，円以外の図形にも適用可能である．本研究では，こうして得られた領域のエッジ点列から，その状態を求め，円の方程式を当てはめることで標識部分の切り出しを行う．

3.3. エッジ点列の状態

先ほど得られたエッジ点列から，その状態を調べる．ここで，状態とはエッジ点列の x 座標と y 座標の増減を表す．その定義を表1に示した．

座標の増減	x↓y↑	x↑y↑	x↑y↓	x↓y↓
状態	0	1	2	3

表1　エッジ点列の状態

この表の意味は，例えば状態 0 であるなら，x 座標が前の座標より減り，かつ y 座標が前の座標より増えた状態を指す．

表のように決めた定義から，通常の円であると，左下から時計回りにトレースしていくので，その状態は 0→1→2→3 となる．そこから，この状態，もしくはこの状態のうち 3 つの状態が連続している場合，その部分に円の方程式を当てはめる．例えば，得られた円の状態の中に，0→1→2 や 2→3→0 が入っている場合である．このように状態を求めることで，標識の赤い部分が一部欠けたり(図 2)，つながってしまっても標識部分を切り出すことができる．図 2 では，標識のエッジの白くなっている部分に対して円の方程式を当てはめている．

3.4. 円の方程式の当てはめ

前節までで得られた領域が円であるかどうか調べる．そこで，トレースを開始する点を原点とする座標系を考える．原点を通る円の方程式は，以下のように与えられる．

$$x^2 + y^2 + 2gx + 2fy = 0$$

(g, f):円の中心座標

エッジ点列から適当な 2 点$(x_1, y_1)(x_2, y_2)$を選び，以下の連立方程式を解く．

$$\begin{cases} x_1^2 + y_1^2 + 2gx_1 + 2fy_1 = 0 \\ x_2^2 + y_2^2 + 2gx_2 + 2fy_2 = 0 \end{cases}$$

それにより，円の中心を求めることができる．また，円の半径 r は，円が原点を通るという制約のため半径 $r = \sqrt{g^2 + f^2}$ となる．(図 3)

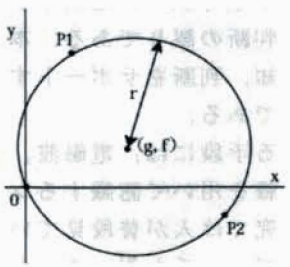


図3　円のパラメータ算出方法

このようにして円のパラメータを求めることができるが，円でないものも無理矢理当てはめることができる．そこで当てはめ誤差について検証した．

円の当てはめ誤差を図 4 のように定義する．

$$E = \frac{1}{n} \sum \left| \frac{r}{q} - \frac{4}{6} \right|^2$$

r:当てはめた円の半径　q:画像上の円の半径

n:エッジ点の総数

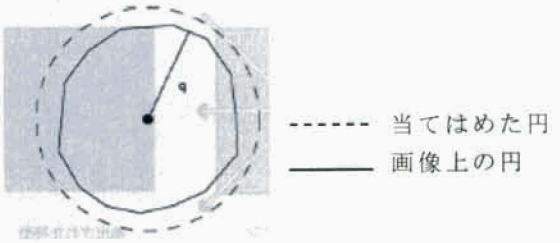


図 4　円の誤差計算方法

この誤差 E が小さいほど，よりよく円に当てはまっているということになる．この値が 0.017 よりも小さい場合，円と判断し標識候補として切り出す．

4. 標識内部のセグメンテーション

本研究では認識に図形の構造的特徴を用いる．よって，標識内部の記号部分と背景にセグメンテーションする必要がある．しかし，これまでに述べた方法では，道路交通標識の外側の赤い部分を切り出すための処理であり，標識内部をセグメンテーションすることができない．そこで，内部をセグメンテーションするために，ヒトの視覚系に学んだ Opponent-Color Filter を開発した．

4.1. Opponent-Color Filter

Opponent-Color Filter とは，ヒトの目の受容野をモデル化したフィルタである．この受容野は，中心に赤い光が当たると発火し，周辺に緑の光が当たると抑制される細胞である．[2]

ヒトの脳の中にある色を知覚する細胞は，三つの錐体からなっている．それらは，L 錐体，M 錐体，S 錐体であり，簡単に言えば，光の三原色である RGB にそれぞれ反応する．これらの受容野は，興奮性，抑制性の領域が同心円状に広がっている．L 錐体から興奮性の反応を受け取る中心部分と，M 錐体から抑制性の反応を受け取る周辺部分を持つ受容野を L-M Color Opponent 細胞と呼ぶ．このような構造を持つ細胞は，道路交通標識内部の背景を構成する大きな面積の白，黄色，青の領域には反応しない．よって，Opponent-Color Filter は，標識内部を背景と記号部分に分けるフィルタとなる．

Opponent-Color 細胞を再現するフィルタは，符号を逆にしたラプラシアンガウシアンフィルタであり，以下の式で表される．

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \text{(ガウシアンフィルタ)}$$

$$F(x,y) = -\nabla^2 G(x,y) = -\left[\frac{1}{\pi\sigma^4} \left(\frac{x^2+y^2}{2\sigma^2} \right) + \frac{1}{\pi\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$F_+(x,y) = F_-(x,y)$$

この式の F+に赤い光を，F−に緑の光を入れる．模式的に表すと下の図 5 のようになる．

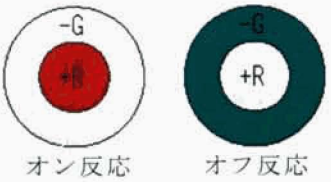


図 5　Opponent-Color Filter 模式図

このフィルタの中心に，赤や白といった光が当たると出力値は大きくなり，逆に青い光が当たると出力は抑えられる．このようにして，切り出された道路標識候補より赤い部分を強調し，青い部分を抑制することで，その内部をセグメンテーションすることができる．実際の画像でフィルタリングした結果が図 6 となる．

4.2. 二値化手順

続いてこのフィルタリングされた画像を二値化する．ここでは，Opponent-Color Filter と判別分析二値化法を用いた二値化処理の流れについて述べる．その手順は以下の通りである．

1．　標識候補として切り出した領域に Opponent-Color Filter をかける．

2．　同じ領域に判別分析二値化法を施す．この際，図の中心の白い領域をマスクとして残しておく．

3．　1 で作った画像に対し，2 で作ったマスク領域の部分にだけ判別分析二値化法を施す．

このような手順で，標識全体の画像から，標識内部を鮮明にすることができる．以下の図 6 がその手順を図示したものである．

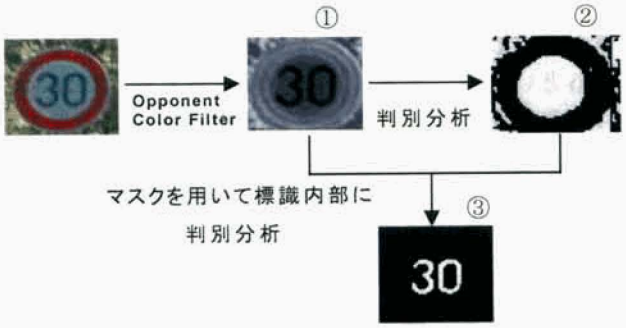


図 7　標識内部の二値化手順

ここで，本研究で提案する Opponent-Color Filter により，フィルタリングした画像(図 7①)の明度ヒストグラムを見てみる．(図 8)

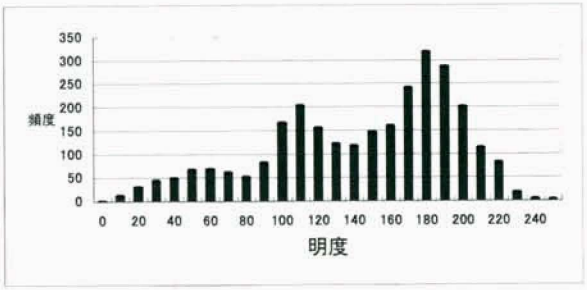


図 8 フィルタリング後のヒストグラム

図 8 では双峰性を見ることができる．ここで，判別分析二値化法を行って二値化すると，図 7②の画像が得られる．この画像から，中心部分を残すマスクを作成する．作成したマスク部分のみの部分の明度のヒストグラムを見ると，以下の図 9 のようになる．

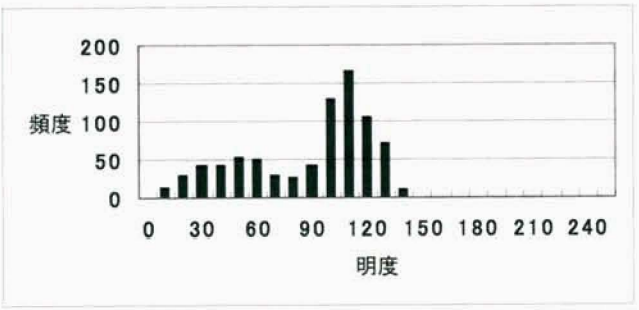


図 9　マスク部分のヒストグラム

図 9 でも同様に双峰性を見ることができ，判別分析二値化法によりフィルタの出力値が低い図の部分と，出力値が高い値の部分に分けることができた．その結果が図 7③の画像になる．

4.3. セグメンテーション結果の比較

ここでは以下の手順で二値化を行った結果と，その他の手法の結果を比較する．比較のための手法として，従来使われてきた Mean Shift Filtering[3]と Bilateral Filtering[4]を用いた．これらの手法はどちらもエッジを保つ平滑化フィルタの一種であり，領域分割手法である．表 2 がそれらの比較である．

表を見てわかるように，本手法の結果は良好なものであるといえる．従来の手法では，ある程度の領域分割は可能であるが，本来必要な標識内部の二値化がうまくいかないことがわかる．しかし，本手法では，標識内部のみをきれいに取り出すことができる．

5. 認識

標識内部の記号部分を抽出した後，どの標識かを認識する．認識は，記号部分の構造的特徴を用いた決定木によって行う．構造的特徴を用いる方法は，標識の大きさや，二値化の程度に対して頑健であり，テンプレートマッチング等の手法に比べて高速に処理できる．

5.1. 駐車禁止・駐停車禁止・U ターン禁止

駐車禁止・駐停車禁止・U ターン禁止の標識については，図 10 のように画像の特定の部分を走査して特徴を抽出した．画像を矢印の方向に走査し，ピクセルの白黒で符号化する．この抽出された特徴は，ここで対象としている標識について一意に定まり，標識の大きさ，二値化の程度に対してロバストである．

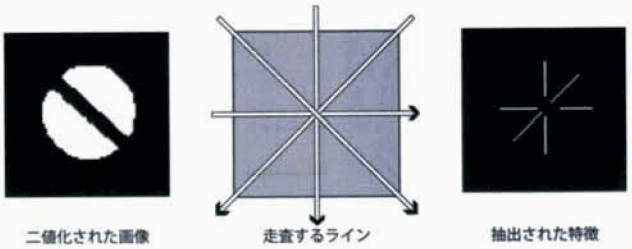


図 10 特徴抽出方法 1

図 10 の中央の図のように走査することで，同じピクセルの連続を符号化する．図 10 の駐車禁止の標識の場合は，次のように符号化される．

横のライン：黒白黒白黒
縦のライン：黒白黒白黒
右上から左下：黒白黒白黒
左上から右下：黒

このように符号化することで，照合が単なる文字列の照合と同じ枠組みになるので，高速に処理ができる．

5.2. 追い越し禁止

追い越し禁止については，画像上で二つの領域に分割され，各領域の面積が大きく異なるという特徴を用いた．二値化画像を X 軸方向に射影したヒストグラムを作り，ヒストグラムの谷を見つける．(図 11)この谷が追い越し禁止標識の左右を分ける点となる．この点から左右の矢印部分の面積を比較して認識した．

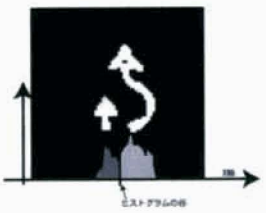


図 11 特徴抽出方法 2

5.3. 速度規制(30km/h, 40km/h, 50km/h)

速度規制標識については，数字の中央を縦に走査し，左右に図があるか無いかという特徴を抽出する．例えば下の図 12 で真ん中の矢印上を走査したとき，それぞれの点の左右に図が存在するかを探す．

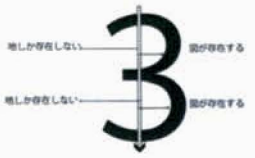


図 12 特徴抽出方法 3

この図のように，右側には図があり，左側には地があるという場合，この文字は“3”であると判別される．

5.4. 決定木

以上のルールから決定木を用いて分類する．決定木は下の図 13 の通りである．本研究で対象としている標識がまだ少ないが，今後同様にして増やしていくことが可能である．

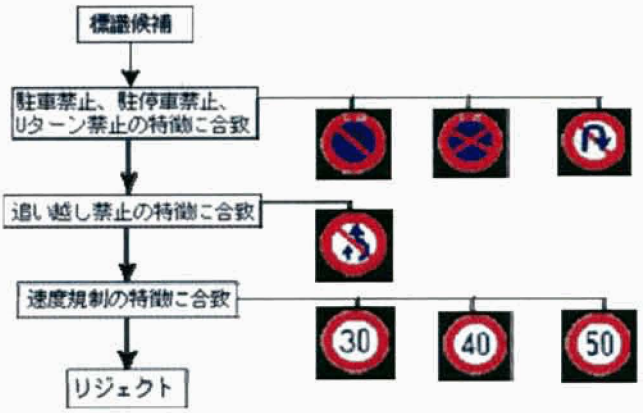


図 13 決定木

6. 実験

以上のアルゴリズムを実装して実験を行った．実験は，ビデオカメラを乗せた三脚を自動車の助手席に固定して撮影し，PC によって処理を行った．本実験では市街地を昼間約 2 時間撮影した．

デジタルビデオカメラは Panasonic(AG-DVX100A)，PC は Dell Precision 340,Intel Pentium4 2.8Ghz,2GB Memory を使用した．IEEE1394 端子にて PC に映像を取り込む．キャプチャする画像のサイズは 720×480 ピクセルのフルカラー画像である．この取り込んだ画像に対して順次アルゴリズムを適用してゆく．カメラのズーム機能を使用しなかった．この画像から切り出される標識のサイズは 20～60 ピクセルの大きさである．

6.1. シーンからの道路標識部分の切り出し率

道路交通標識を認識するために，映像から道路交通標識候補を画像より切り出す．その性能を評価するために切り出し率を用いる．切り出し率とは映像に含まれる標識の数に対する，本研究のアルゴリズムにより映像から切り出された標識の割合である．結果は以下の表 3 のようになった．

全ての標識の数	切り出した標識の数	切り出し率
２３５	２１４	０．９１

表 3 切り出し率の結果

また，誤って切り出してしまった標識候補は 14 個あった．

6.2. 処理速度

ここでは，本研究の課題であるリアルタイム性についての結果について述べる．1 フレームあたり，標識を切り出して認識するまでにかかる時間は，標識 3 個を含むフレームで約 120ms であった．よって，実時間で認識処理が可能であるといえる．

6.3. 認識結果

最後に，切り出しおよびセグメンテーションした画像についての認識結果を示す．この認識結果は，実験に用いたシーン映像の各フレームから標識を切り出し，その切り出した標識についての認識率となっている．本アルゴリズムでは，一つの標識に対して複数のフレームにわたって切り出される．そうした複数のフレームの中で最後のフレームに対して，全 214 個の標識中 206 個が正解，8 個が誤って認識された．認識率は 96% となった．また，誤って標識候補として切り出してしまったものに対しては，14 個中 7 個が正しくリジェクトされた．

7. 結果の考察と今後の課題

7.1. 結果の考察

シーン中から標識候補を切り出す際，うまくいかなかった原因の一つは，標識の背景に赤い看板等があった場合である．その例を図 14 に示した．左側が実際に撮影された画像であり，右側が色相によりラベル付けした画像である．これは，今のアルゴリズムでは解決できないので，新たな手法を考える必要がある．また，失敗のもう一つの原因は標識が反対車線にあって，画像上の標識の大きさが小さい，または形がゆがんでしまった場合である．しかし，反対車線にある標識は，自車線にも同じ標識が出現する場合がほとんどで，重要性は低いといえる．

切り出した標識の認識率は，96%とおおむね良好な結果であるといえる．しかし，光の当たり方で人間の目には普通に見えても，ビデオカメラを通した画像であると反射して標識内部がわからなくなる場合がある．この例を図 15 に示した．また，標識自体が傾いてしまっても，あまりうまくいかないことがわかった．

7.2. 今後の課題

今後の課題は，上で述べたこと以外に，対象とする標識を増やすことがあげられる．現在は 7 種類で行っているが，これでは少ない．実用的には，さらに多くの標識を対象とする必要がある．また，今回は昼間の市街地を撮影した映像で実験を行ったが，夕方や雨の日等，条件が違うと結果も変わってくると考えられる．このような条件においても，今後考察する必要がある．

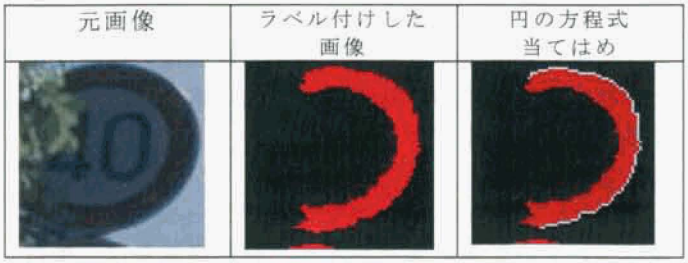


図 2 欠けている円の切り出し

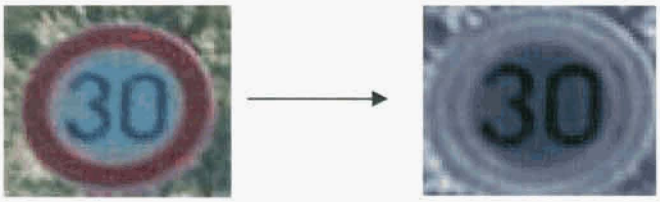


図 6 Opponent-Color Filter をかけた画像

表 2 各手法の比較

図 14 切り出せなかった標識の例

図 15 認識できなかった標識の例

文 献

[1] 国土交通省 ITS ホームページ
http://www.mlit.go.jp/road/ITS/j-html/

[2] 平井有三, "視覚と記憶の情報処理", 倍風館, 東京, 1995

[3] D.Comaniciu and P.Meer."Mean Shift : A robust approach toward feature space analysis", IEEE Trans. On Pattern Analysis and Machine Intelligence, PAMI-14(5):603-619,2002

[4] C.Tomasi and R.Manduchi. "Bilateral filtering for gray and color images.",In ICCV, page839-846,1994

[5] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification(2nd Edition). John Wiley & Sons, Inc., New York, 2001.

[6] D. M. Gavrila. Traffic sign recognition revisited. In Proceedings of the 21st DAGM Symposium for Musterekennung., pages 86-93. Springer Verlag, 1999.

[7] H.A.Glucksman. Classification of mixed font alphabetics by characteristic loci. In Digest of 1st Ann. IEEE Computer Conference, pages 137-141,1967.

[8] G. Piccioli, E. De Micheli, P. Parodi, and M.Campani. Robust method for road sign detection and recognition. Image and Vision Computing, 14:209-223, 1996.

[9] L. Spillmann and J. S. Werner. Visual Perception: The Neurophysioligical Foundation. Academic Press, San Diego, California, 1990.

pp.939-946

反対色フィルタを用いた道路交通標識の実時間認識

馬場今日子† 平井有三††

† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士前期課程
つくば市天王台 1-1-1
†† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
つくば市天王台 1-1-1
E-mail: †baba@viplab.is.tsukuba.ac.jp, ††hirai@is.tsukuba.ac.jp

あらまし 安全運転支援を目的に、車載カラービデオカメラを用いた実時間道路交通標識認識システムを開発している。本報告では、赤の輪郭を持った円形の標識 6 種を対象として構成した実験系について報告する。認識系は、(1)HSV 色空間と円の方程式の当てはめによる赤い円形の対象の検出処理、(2) 反対色フィルタと判別分析法による円内部のシンボル部の切り出しと識別処理からなる。シンボル部の切り出しには、人の視覚系にある赤ー緑反対色フィルタを使用した。約 2 時間撮影した道路映像を用いて性能を評価した。その結果、95%を超える認識率が得られた。また、識別対象とした標識の配色は、人の視覚系の情報処理様式に適合しているという示唆を得ることができた。
キーワード ITS、安全運転支援、道路交通標識、反対色フィルタ、HSV 色空間

A Real-time Recognition System of Road Sign Symbols using an Opponent-color Filter.

Kyoko BABA† and Yuzo HIRAI††

† Graduate School of Systems and Information Engineering, Major of Computer Science, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
†† Graduate School of Systems and Information Engineering, Deartment of Computer Science, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
E-mail: †baba@viplab.is.tsukuba.ac.jp, ††hirai@is.tsukuba.ac.jp

Abstract In order to support safe driving, a real-time recognition system for road traffic signs are developing by using an in-vehicle video camera. In this paper an experimental system focused on six circular signs with red annular boundaries is described. The recognition system consists of (1) detection of red annular objects by using HSV color space and by matching a circle equation to the object boundary, and (2) segmentation of sign simbols by using an opponent-color filter and discriminant analysis, and identification of symbols by structural features and a decision tree. Red-green opponent-color filter found in our visual systems was used for the segmentation of symbol parts. The performance of the system has been evaluated by using about 2-hour video images taken in the daytime. Recognition rate was more than 95%. Besides, it is also suggested that the adequateness of the combinations of colors in road signs may be justified by the function of our visual system.
Key words ITS, safe-driving support, road traffic signs, opponent-color filter, HSV color space

1. はじめに

平成 15 年に国内で発生した交通事故件数は約 95 万件で、7,700 人が死亡し、118 万人が負傷している [1]。発生件数の約 6 割を追突事故と出会いがしらの事故が占めている。死亡者数は昭和 45 年の 16,000 人をピークに減少傾向にあるが、発生件数は車の保有台数に比例して増加し続けている。米国でも、2002 年に 680 万件の事故で 42,000 人が亡くなっている。

この様な背景の下、世界でITS(Intelligent Transport System)に関する研究が1990年頃から盛んに行われてきた。国内では高度道路交通システム(ITS)[2]の一環として、自動車運転の安全と円滑さを目標に走行支援道路システム(AHS:Advanced cruise-assist Highway System)[3]に関する研究が多くの研究機関で行われている。

AHSにはAHS-i、AHS-c、AHS-aの三つのレベルがあるとされる。AHS-iは情報収集の一部をシステムがサポートするものであり、前方障害物情報の表示板による提供など現在実用化が最も進んでいるレベルである。AHS-cは運転操作の一部をシステムがサポートするレベルであり、AHS-aは完全自動運転のレベルである。

AHS-iにおける情報収集にも色々な対象が考えられる。現在実用化されている前方障害物情報は数kmのレンジであり、運転者からは見えない情報である。しかしながら、平成15年の事故統計[1]によれば、安全不確認、脇見運転、動静不注視などの事故原因が全体の55%を占めている。したがって、運転者から見える情報を見落とさないように支援することも重要な課題である。

本研究の目的は、運転者から見える情報の見落とし防止を支援する技術を開発することにある。道路交通標識は、運転者にとって最も身近な見える対象であり、かつその見落としが直接大きな事故につながる危険性がある。道路交通標識を車載ビデオカメラを用いて認識しようとする試みは、1990年代から海外[4][5][6][7][8]や国内[9][10]で活発に行われている。

ビデオカメラを用いた道路交通標識認識は、どの研究でもおおよそ以下の手順に従って行われている。

（1） 色情報を用いた標識部の識別
（2） 幾何学的情報を用いた標識部の切り出し
（3） 標識内部のシンボル部分の切り出し
（4） シンボル部分の識別

色による標識部の識別には、RGB表色系から色相と彩度と明度によるHSV表色系へ変換し、主として色相を手がかりに識別を行うものがある[11][12][13]。しかしながら、HSV表色系は彩度が低いところで色相が不安定になることから[14]、RGB減算法を用いたもの[15]や、RGB間の比を用いたもの[5]が提案されている。また、色相は照明条件や距離や年月によって変化することから、濃淡画像のみを使用するもの[9][6][16]などが提案されている。

幾何学的情報を用いた標識部の切り出しには、エッジの勾配方向へ引いた直線の交点から円の中心を求めるもの[11]、距離変換を用いたもの[6]、図形の対称性を用いたもの[17][13]などがある。また、色と形の情報を用いてエネルギーを定義し、GA(Genetic Algorithm)やSA(Simulated Annealing)により最適化問題を解いて交通標識を検出するものなどが提案されている[9][18][19]。

標識内部のシンボル部の切り出しには、エッジを用いたもの[11]、背景とシンボル部の輝度差を利用したもの[13]などが提案されている。また、シンボル部の識別には、正規化相互相関を用いたもの[11]、誤差逆伝搬型ニューラルネットワークを用いたもの[19]、水平・垂直形状特徴量を用いたもの[13]などがある。

図1 本報告で対象とした道路交通標識。

本研究では、HSV表色系を用いてシーン中の赤い標識候補をラベル付けしている。幾何学的情報による標識部の切り出しは、ラベル付けされた領域の輪郭線に円の方程式を当てはめ、その誤差がしきい値以内の対象を標識候補とした。また、標識内部のシンボル部の切り出しは、人の視覚系にみられる赤－緑反対色フィルタと判別分析法を組み合わせて行った。シンボルの識別は、構造的な特徴の抽出と決定木により行った。システムをPC上に実装し、性能評価を行った。図1に本論文で対象とした道路交通標識を示した。

本論文の構成は以下の通りである。第2節で色情報を用いた標識部の識別法と、円の方程式の当てはめによる標識候補の切り出しについて述べる。第3節で反対色フィルタを用いたシンボル部の切り出しについて述べる。第4節で識別処理について述べる。第5節で実験結果を紹介し検討を加える。第6節で本論文をまとめる。

2. 道路標識候補の切り出し

道路交通標識の中で規制標識の配色は、赤、白、青の組み合わせで構成されている。したがって、色を手がかりにすれば、道路画像から画素数に比例した時間で標識候補部分を切り出すことが可能になる。本研究では、他の多くの研究でも利用されているHSV色空間を用いて標識部を識別する。

2.1 HSV表色系を用いた標識部のラベル付け

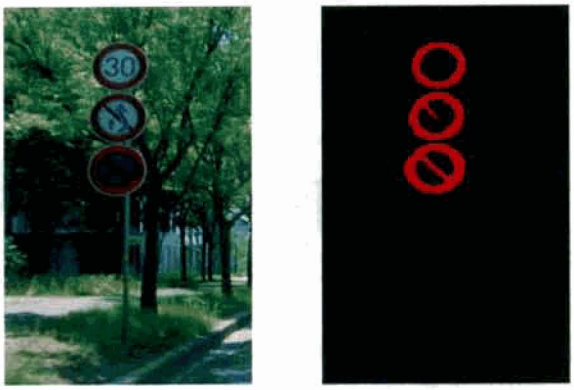
色相(Hue)、彩度(Saturation)と明度(ValueまたはBrightness)は、物体表面色を規定したマンセル表色系の色を表す次元である。多くの場合、RGB表色系から物体の色を直接想像することが困難であるのに対し、HSV表色系は人間の感覚にあった色表現を可能にする。RGB表色系からHSV表色系への変換は、RGB表色系におけるそれぞれの要素の値を、R、G、Bと表現し、HSV表色系における色相をH、彩度をS、明度をVと表現すれば以下のようになる[20]。

$$MAX = \max\{R,G,B\}, \quad MIN = \min\{R,G,B\}$$

$$H = \begin{cases} \left(0 + \frac{G-B}{MAX-MIN}\right) \times 60 & \text{if } R = MAX \\ \left(2 + \frac{B-R}{MAX-MIN}\right) \times 60 & \text{if } G = MAX \\ \left(4 + \frac{R-G}{MAX-MIN}\right) \times 60 & \text{if } B = MAX \end{cases} \quad (1)$$

$$S = \frac{MAX - MIN}{MAX} \quad (2)$$

$$V = MAX \quad (3)$$



(a) (b)

図 2　原画像 (720 × 480) の一部 (a) と赤の画素にラベル付けした処理結果 (b)

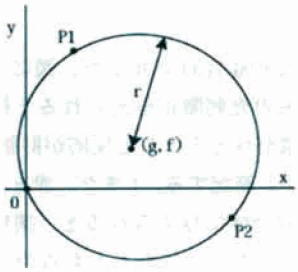


図 3　円の方程式を求める 3 点アルゴリズム

H は 0 から 360 度まで変化し、360 度 = 0 度とする。本研究では、赤い円形の標識を検出対象としているので、

$$324 \leqq H \leqq 28.8, \quad S \geqq 0.2, \quad V \geqq 0.1 \tag{4}$$

となる画素を、標識の赤い部分として「赤」にラベル付けする。彩度が低い部分の色相は不安定になるので、判断していない [14]。実画像とラベル付けされた画像の例を図 2 に示した。画像のサイズは 720 × 480 画素である。

2.2　円形領域の切り出し

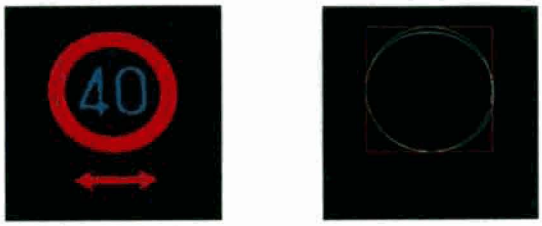
本論文で対象としている標識は赤い円形の標識なので、前節で「赤」にラベル付けされた画像の中から輪郭が円形の対象を検出し、標識候補とする。円形であるか否かの判定は、対象のエッジをトレースした輪郭線上の 3 点から円の方程式を求め、輪郭線と求めた円との誤差を評価して決める。図 3 に輪郭線上の 3 点を用いて円の方程式を求める様子を示した。3 点のうち 1 点を座標原点上にとれば、円の方程式は

$$x^2 + y^2 - 2gx - 2fy = 0 \tag{5}$$

となる。(g, f) は円の中心座標であり、求める円のパラメータである。残りの 2 点 $p_1 = (x_1, y_1)$ と $p_2 = (x_2, y_2)$ から、円のパラメータは

$$\begin{cases} x_1^2 + y_1^2 - 2gx_1 - 2fy_1 = 0 \\ x_2^2 + y_2^2 - 2gx_2 - 2fy_2 = 0 \end{cases} \tag{6}$$

を解くことで求めることができる。円の半径 r は、$r = \sqrt{g^2 + f^2}$ である。



(a) (b)

図 4　(a) 赤でラベル付けされた標識候補画像。分かりやすくするために、文字部の青もラベル付けしている。(b) 実際の輪郭 (内側の線) と当てはめた円。

求めた円の方程式と輪郭線上のすべての点との誤差を評価し、円であるか否かの判定を行った。輪郭線上の i 番目の点と円の中心までの距離を q_i、点の総数を N とすれば、誤差 E は

$$E = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{r}{q_i}\right)^2 \tag{7}$$

で評価する。実験により、$E \leqq 0.015$ の場合、円と判断した。図 4 に当てはめた例を示した。

道路交通標識は、運転席から常に正面に見えるわけではないので、楕円の当てはめの方が精度よく標識候補を切り出すことができる。文献 [21] のコンテストで使用された道路標識の写真データを用いて予備的に比較したところ、楕円当てはめの精度は 82.2%で、円の当てはめより 3%程度よかった。しかしながら、処理時間は円の当てはめが 100msec、楕円の当てはめが 250msec 程度かかった。実時間処理を優先するため、本研究では円の当てはめを用いた。

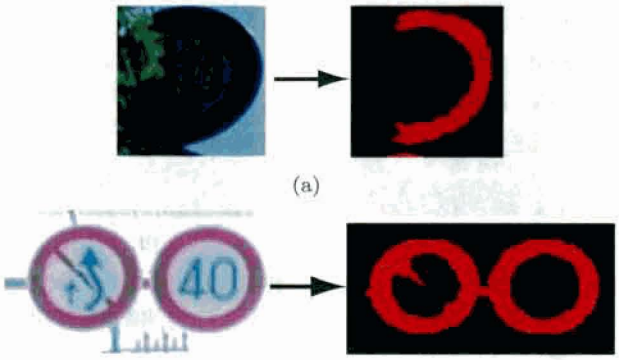
2.3　円の補完と分節

道路交通標識は、見通しの良いところだけに立てられているわけではなく、木や電信柱により部分的に隠蔽されている場合もある。図 5(a) は木により一部が隠されている例であり、「赤」でラベル付けした結果が右の図である。また、ビデオカメラの色処理も不完全であり、にじみなどの原因により図 (b) のように二つの標識がつながってしまう場合がある。認識精度を上げるためには、このような状況下でも、図 (a) のような場合は一つの標識、図 (b) のような場合は二つの標識候補として切り出せることが必要である。そのために、円の方程式を当てはめるべき輪郭線の部分系列を同定することが必要となる。

円の方程式を当てはめてよい輪郭線の部分系列の同定は、エッジを時計回りにトレースするときの座標値の増減により状態を定義し、その状態系列で判断する。すなわち、エッジ点列の状態を下記のように定義する。

座標値の増減	$x \downarrow y \uparrow$	$x \uparrow y \uparrow$	$x \uparrow y \downarrow$	$x \downarrow y \downarrow$
状態	0	1	2	3

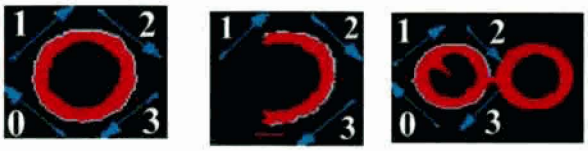
図 6(a) に示したように、通常の円の場合、状態系列は $0 \rightarrow 1 \rightarrow 2 \rightarrow 3$ のように推移する。この場合、状態 3 と状態 0 はつながっているものと考え、系列の開始位置はどこにあってもよいものとする。図 5(a) のように円が不完全な場合に対応するため、円を構成する 4 つの状態の系列のうち、3 つ以



(a)

(b)

図 5　(a) 木によって一部が隠された標識 (左) と赤でラベル付けした結果 (右)。(b) ビデオカメラの色のにじみにより、二つの標識 (左) が赤でラベル付けするとつながってしまった例 (右)。



(a)　(b)　(c)

図 6　(a) 完全な円の状態系列。(b) 図 5(a) の場合、部分系列 1 → 2 → 3 があるので、円の方程式を当てはめる。(c) 図 5(b) の場合、円の系列 3 → 0 → 1 → 2 がある。

上の状態が連続して現れる場合には、そのエッジ点列に対して円の方程式を当てはめてよいと判断する。したがって、図 6(b) に示されているように、隠蔽によって一部が欠けた円の場合にも標識候補として切り出すことができる。また、図 (c) のように、二つの標識がつながってしまっている場合も、円を構成する状態系列 3 → 0 → 1 → 2 があるので、それぞれを一つの標識候補として切り出すことができる。このような処理で、切り出し率を 5%以上上げることができた。

3.　標識内部の記号の切り出し

切り出された道路交通標識を識別するためには、標識内部の記号を識別する必要がある。内部の記号を切り出す処理を行わず、標識全体の大きさを正規化し、形が似たすべての鋳型との画像ベースの正規化相互相関により識別する方法も提案されているが [19]、車の移動により標識の大きさは常に変化するので大きさに依存しない認識手法の方が望ましい。また、標識内部の記号の形やそれらの相対的位置は、地域ごとに異なっており、これらの変動を吸収できる認識手法の方が望ましい。したがって、本研究では標識内部の記号部分を切り出し、構造的特徴を用いた認識手法を用いた。

記号部分の切り出しには、近傍間での色差を検出すると同時にノイズ除去機能も有する、反対色フィルタによる切り出し手法を提案する。

3.1　反対色フィルタ

反対色フィルタは、ほ乳類の網膜神経節細胞や外側膝状体と呼ばれる中継核にみられる反対色細胞に相当する。代表的な反対色細胞の一つである赤－緑反対色細胞の受容野の例と色光

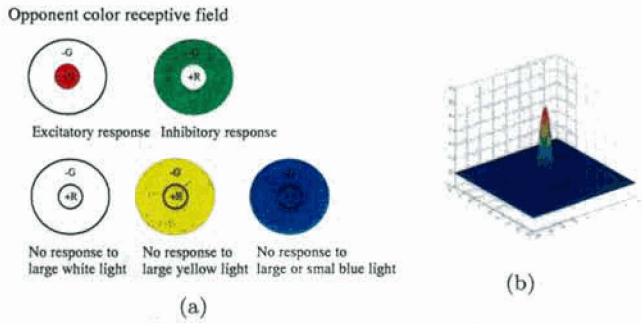


図 7　(a) 赤－緑反対色受容野を持つ神経節細胞の色刺激に対する反応特性。受容野の中心に赤い光が当たると興奮し (上左)、周辺に緑の光が当たると活動が抑制される (上右)。白色光や黄色、青の光が受容や全体に当たると興奮と抑制がキャンセルして反応しない (下左、下中、下右)。(b) 反対色フィルタの Laplacian Gaussian 関数モデル。

に対する反応特性を図 7(a) に示した。図に示したように、受容野の中心に赤色の光刺激が与えられると細胞は反応し、周辺部に緑の光刺激が与えられると反応が抑制される。(このような細胞を+R-G と略記する。) また、赤と緑を含む白色光や黄色の光が受容や全体に与えられると、興奮と抑制がキャンセルしあって反応しない。また、どちらの光も含まない青に対しても反応しない。赤－緑反対色細胞には、+R-G 型以外に、-R+G、+G-R、-G+R 型がある。赤－緑型以外には、青－黄型、白－黒型がある。

このような受容野の空間的な広がりを表すモデルとして、ガウス関数を 2 回微分した Laplacian-Gaussian 関数がよく使われる。すなわち、

$$G(x,y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{8}$$

$$F(x,y) = -\nabla^2 G(x,y) = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{9}$$

$$F^+(x,y) = \begin{cases} F(x,y) & \text{if } F(x,y) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

$$F^-(x,y) = \begin{cases} 0 & \text{if } F(x,y) > 0, \\ F(x,y) & \text{otherwise.} \end{cases} \tag{11}$$

で定義される。関数の外形を図 7(b) に示した。

RGB 表色系で表された入力画像の位置 (x,y) における各色要素を $R(x,y)$、$G(x,y)$、$B(x,y)$ とすれば、+R-G 型の反対色フィルタの出力は、

$$O(x,y) = \int\int F^+(\xi,\eta) R(\xi - x, \eta - y) d\xi d\eta + \int\int F^-(\xi,\eta) G(\xi - x, \eta - y) d\xi d\eta \tag{12}$$

と表現される。ガウスフィルタは平滑化フィルタとして知られており、赤、緑成分のノイズを除去しながら近傍間の赤－緑成分間の差を検出していると考えることができる。また、近傍が同じ色であれば、画像の赤い部分からは正の出力、緑の部分からは負の出力、白い部分からは 0 付近の出力が得られることになる。

フィルタの空間的な広がりを決めるパラメータ σ は、標識の円の直径 r が、$r < 30$ 画素の場合 0.75 画素、$30 \leqq r < 40$ 画

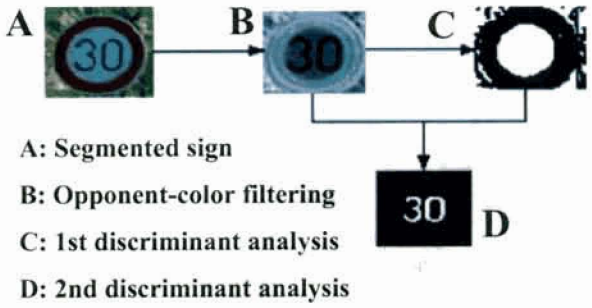


図 8　反対色フィルタと判別分析 2 値化法を用いたシンボル部の抽出。

素の場合 1.0、$40 \leqq r < 50$ 画素の場合 1.25、$50 \leqq r < 60$ の場合 1.5、$r \geqq 60$ の場合 1.75 画素に実験により設定した。

3.2　判別分析 2 値化法によるシンボル抽出

判別分析 2 値化法は、画像の濃度ヒストグラムに判別分析法を適用し、濃度の明るいクラスと暗いクラスの識別境界を求め画像を 2 値化する手法である。図 8 に判別分析 2 値化法を用いたシンボル抽出法を示した。切り出された標識候補 (A) に赤緑反対色フィルタをかけた結果が B の画像である。この画像の出力値のヒストグラムを図 9(a) に示した。−2,905 から 3,270 の範囲の出力値を、1 から 256 の範囲に正規化している。標識の赤い部分が大きな出力値側に、文字の背景となっている白い部分が出力値 0(横軸の 120 に対応) の周辺に、青い文字部が低い出力値側に分布している。

判別分析法は、クラスの識別境界を、クラス内分散とクラス間分散の比を最大にする出力値をしきい値とする。出力値 t を変数として評価関数を書けば、

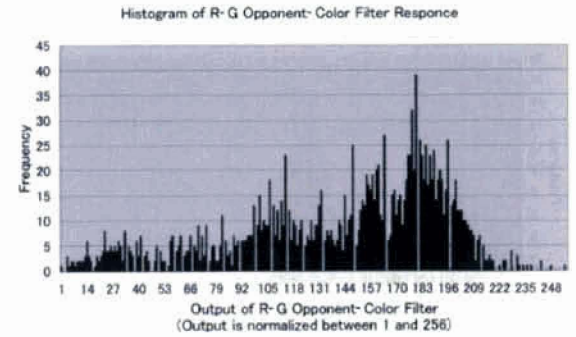
$$f(t) = \frac{\text{クラス間分散}}{\text{クラス内分散}} = \frac{P(C_1)(\eta_1 - \bar{\eta})^2 + P(C_2)(\eta_2 - \bar{\eta})^2}{P(C_1)\sigma_1^2 + P(C_2)\sigma_2^2} \quad (13)$$

となる。ここで、$P(C_i), i = 1, 2$ はクラス i の比率、η_i はクラス内平均、$\bar{\eta}$ は全平均、σ_i^2 はクラス内分散である。

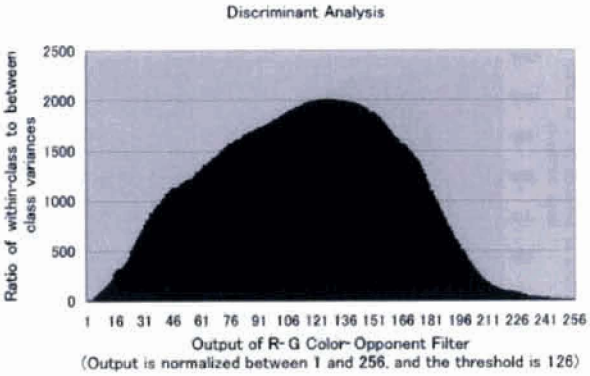
図 9(a) のヒストグラムに対して、式 (13) で求めた評価関数を図 (b) に示した。評価関数が最大値を示す出力値を 2 値化のためのしきい値とすればよい。この場合、しきい値は 126(実際の出力値は 135) となった。赤い部分と、白や青の部分に分けられることになる。図 8 の C 図に 2 値化した結果を、赤側を黒、白側を白で示した。

判別分析 2 値化法は、2 クラス問題の識別境界を求める手法である。図 8 の標識は、赤、白、青で構成されており、シンボル部分の青を抽出するためには、白側のクラスをもう一度白と青に分類する必要がある。図 8 の C 図で得られた中央の円形部分に対応する領域のみに、再び判別分析 2 値化法を適用すれば、背景の白とシンボル部の青を識別することができる。図 10(a) に中央部の反対色フィルタ出力値のヒストグラムを示した。判別分析法の評価関数を図 (b) に示した。得られたしきい値は 68(出力値は−1265) で、2 値化した結果を図 8 の D 図に示した。標識のシンボル部分が抽出できていることがわかる。

判別分析 2 値化法を 2 回適用してシンボル部分を抽出する方法は、図 11(a) に示したような赤と青でできている標識に対しても有効である。赤と青が混じり合ったファジーな 3 番目のクラスが存在すると考えられる。図 (b) に判別分析 2 値化法を 1 回適用した場合を、図 (c) に 2 回適用した場合を示した。後段の標識識別器にとっては、明らかに 2 回適用した方が処理が楽である。



(a)



(b)

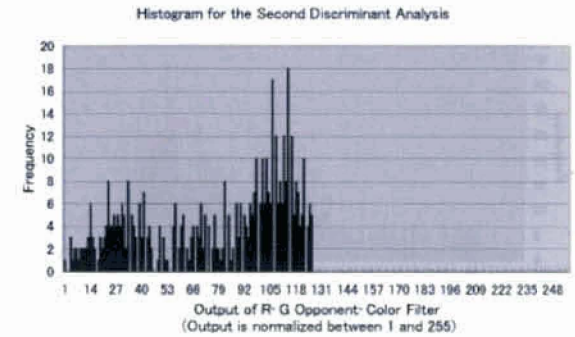
図 9　(a) 切り出された標識候補 (図 8 中の A) に赤緑反対色フィルタをかけた画像 (同図 (B)) の出力値のヒストグラム。−2,905 から 3,270 の範囲の出力値を 1 から 256 の間に正規化している。出力値 0 は 120 に対応している。(b)2 値化のしきい値を各出力値に設定したときの判別分析法の評価関数値の変化の様子。最大値のところの出力値を 2 値化のしきい値とする。この場合、126(出力値は 134 に対応) がしきい値となる。

4.　道路交通標識の認識

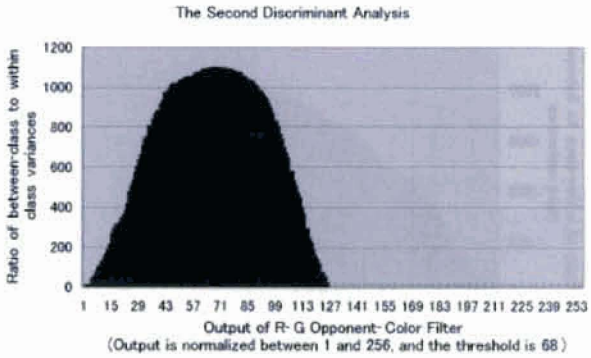
道路交通標識の認識は、標識内部のシンボル部の構造的特徴を抽出し、決定木によって識別することにより行う。この過程を図 12 に示した。駐停車禁止は、反対色フィルターによって抽出されたシンボル部の対称性の有無によって、駐車禁止は、4 方向の方向特徴の有無によって識別している。追い越し禁止は垂直射影した画素数の分布に見られる中央部のくびれを用いて識別している。速度規制の数字は、垂直射影に二つの分離した分布があることを手がかりとして類別し、3、4、5 の識別は交差特徴抽出法 [22] として知られている簡便な構造的特徴を用いて行っている。いずれの標識にも分類されなかった場合は、リジェクトとしている。

5.　実映像による性能評価

以上のシステムを、VisualStudio.Net 2003 上で C++を用いて実装した。赤い物体検出や円検出などの処理モジュールを dll として作成し、(a) 各処理段階におけるパラメータ調整など



(c)



(d)

図 10　(a)1 回目の判別分析 2 値化法で得られた、図 8C の中央円形部分の反対色フィルタ出力値ヒストグラム。(b) 中央部に対する判別分析法の評価関数。最大値は 68(出力値は–1265) のところ。

(a)　(b)　(c)

図 11　(a) 赤と青で構成された標識の例。(b)1 回の判別分析 2 値化法で得られた 2 値化画像。(c)2 回の判別分析 2 値化法で得られた 2 値化画像。

を同じ画面上で行えるようにした実験システム、(b) 切り出された標識候補画像を読み込んで認識率などの性能評価を行うための評価システム、(c) 実際に車に搭載して実時間で道路交通標識を認識するシステムを開発した。ここでは、ビデオカメラ (Panasonic AG-DVX100A、レンズの焦点距離は 35mm 換算で 32.5mm) により市街地を昼間約 2 時間撮影した映像を、PC(Dell Precision 340, 2.8GHz, 2GB メモリ) に一旦取り込み、ビデオレートで再生しながら解析した結果を示す。

1 フレーム中に 3 個の標識がある場合の切り出しに要する時間は約 83 ミリ秒、それらの認識に要する時間は約 37 ミリ秒であった。標識が 1 つの場合、全体の処理時間は約 70 ミリ秒であった。したがって、平均すれば約 10 画像/秒程度で映像がサンプリングできていることになる。

5.1 実験結果

図 13 に切り出された標識画像系列と識別結果の例を示した。速度制限 40Km の場合は最初の 2 枚と 5 枚目で、追い越し禁止の場合は 5 枚目で識別誤りが生じている。一般的な傾向として、図 (a) のように遠くて小さな画像の場合に誤る確率が高くなっている。また、系列の最後の画像は距離が一番近いところにあるので大きく明確な画像が得られ認識率も高くなる。評価法としては、個々に切り出された画像 (例えば図 (a) の 7 枚の画像) の正解数を用いた評価と、各系列の最後の画像 (例えば図 (a) の系列の最後の画像) の正解数を用いた。後者は、対象となっている標識が正しく識別できたか否かの指標として用いた。

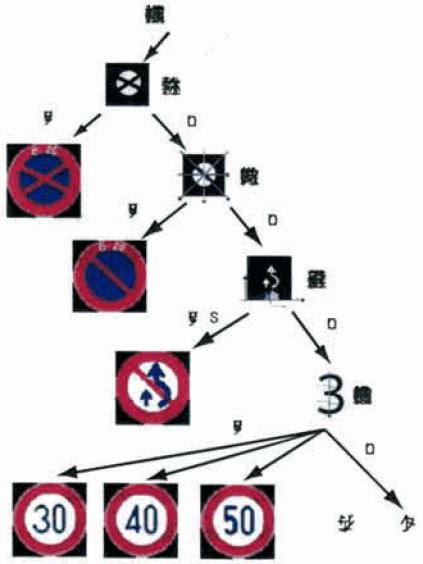


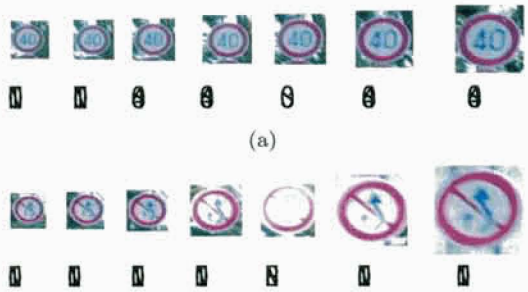
図 12　識別決定木。速度規制の構造的特徴は、30Km のみを示している。詳しくは本文を参照のこと。

約 2 時間の映像中に存在した標識の総数は 212 個で、そのうちの 200 個の標識を標識候補として切り出すことができた (切り出し率 94.3%)。200 個の標識系列には、975 枚の個別画像が含まれていた。そのうち、832 枚を正しく識別することができた (個別認識率：85.3%)。また、200 個の標識系列のうち、191 個の系列で最後の画像を正しく識別することができた (系列認識率：95.5%)。また、標識以外の対象を標識候補として切り出した例は 14 あった。正しくリジェクトできたのは 2 例のみであったが、リジェクトできなかったものの中に系列長が 1 のものが 7 例あった。標識の系列長はすべて 2 以上なので、これらは排除可能と考えている。

現在のところ研究機関間で共通に利用可能な道路映像データベースがないので、他の研究成果との直接の比較はできないが、少なくとも同じような性能は得られているものと思われる。

5.2 標識毎の認識率の比較

表 1 に標識毎の認識率を示した。個別正解数は、小さな画像から大きな画像まで混在しているので 85%程度の正解率にとどまっているが、系列最後の画像に対する正解率は 95%を超えている。速度制限 40Km と 50Km に対する系列認識率が他と比較して悪くなっている原因は、文字部の特徴抽出に簡便な交差特徴抽出法を用いていることにあるものと考えられる。例えば、文字「4」を構成する斜めの線は、2 値化すると途切れてしまうことがよく見られたが、交差特徴抽出法で特定の位置の交差情報を検出するときこの途切れ部分にかかると正しく識別で



(a)

(b)

図 13　サンプリングされた切り出された標識画像と認識結果の例。(a) 速度制限 40Km の例。7 枚の画像を切り出すことができている。各画像の下には、識別結果が示されている。NOT:No Overtaking、S40:速度制限 40Km。(b) 追い越し禁止の例。この場合も 7 枚の画像が切り出せている。NPS:No Parking and Stopping。

表 1　標識毎の識別率 (赤－緑反対色フィルタを用いた場合)

	個別画像数	個別正解数	個別認識率	標識系列数	系列正解数	系列認識率
追い越し禁止	217	198	91.2	47	46	97.9
駐車禁止	442	373	84.4	84	81	96.4
駐停車禁止	12	10	83.3	4	4	100
速度制限 30Km	27	24	88.9	6	6	100
速度制限 40Km	129	102	79.1	26	24	92.3
速度制限 50Km	148	125	84.5	33	30	90.9
合計	975	832	85.3	200	191	95.5

きなくなる。

デザインが他に比べて単純な駐車禁止は系列認識率は 96%を超えているが、個別認識率は 84%程度にとどまっている。誤認識は標識が遠くにあって小さい場合に多く見られる。そのような場合、標識の青色の部分が 2 値化によって面積が小さくなり、図 12 の 4 方向特徴を用いた交差特徴抽出法では特徴抽出がうまくできないことがその原因と考えている。本報告の交差特徴抽出法にはまだ改善の余地がある。

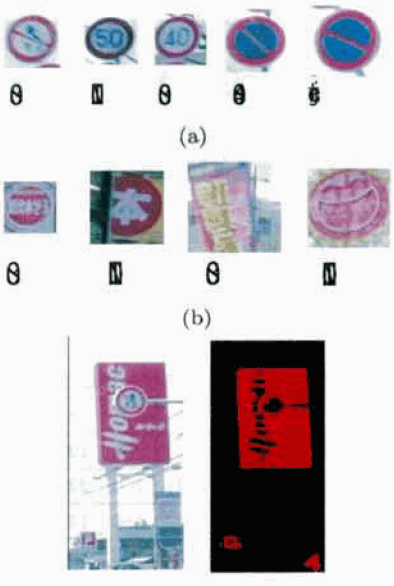
5.3　誤識別の例

誤識別は、標識を別の標識あるいは標識でないと判断したもの (miss)、標識でないものを標識と識別したもの (false alarm)、標識を切り出すことができなかったものに分類できる。これらの例を図 14 に示した。図 (a) に示した miss の画像は、いずれも系列の最後の画像である。標識の傾きや輝点ノイズが誤識別の原因となっている。傾きが原因の場合系列のすべての画像が誤識別になるので、個別識別率が低下する大きな要因となる。図 (b) は false alarm の例である。左の 3 つは系列長が 1 の画像である。正しい標識の場合、系列長は 2 以上なので、長さ 1 の系列を除外することで誤りを避けることができると考えている。一番右の画像は、15 個の画像からなる系列の最後の画像である。図 (c) は切り出せなかった標識の例である。他に、標識が小さすぎる場合にも切り出すことができていない。

5.4　赤－緑反対色フィルタは最適？

本論文で対象にしている標識は、白地に青の数字や、青地に赤の斜め線など、赤と青がどの標識でも使用されている。赤と青は反対色の関係にはないが、赤－青フィルタの方が工学的に最適のように思われる。この点を確かめるために、式 (12) で定義されたフィルタの負の部分に、緑の代わりに青を入力したフィルタを用いて認識率を評価した。表 2 に示した結果から、青地に赤線で構成された駐車禁止の場合、個別認識率は赤－緑反対色フィルタに比べて 5%程度よくなっている。しかしながら、追い越し禁止の場合は、逆に赤－緑反対色フィルタに比べて 5%程度悪くなっており、さらに、系列認識率では 17%も悪くなっている。また、全体の系列認識率も 5%悪い。

この原因を明らかにするために、赤－緑フィルタで認識でき、赤－青フィルタでリジェクトされた追い越し禁止の例を調べた。図 15(a) に切り出された原画像を、(b) に赤－緑反対色フィルタで 2 値化した結果を、(c) に赤－青フィルタで 2 値化した結果を示した。赤－青フィルタでは、右矢印下部が抽出できておらず、リジェクトの原因となっている。この理由を検討するために、右側矢印上部と下部とその周辺部の RGB 成分 (各 10 点の平均値) を表 3 に示した。この表から、B 成分は図の部分でも地の部分でも大きな値を示しているのに対して、R、G 成分は図と地の部分で大きな差があることがわかる。



(a)

(b)

(c)

図 14　誤識別の例。(a)miss の例。いずれも系列の最後の画像。(b)false alarm の例。左の 3 つは系列長が 1 の画像。(c) 切り出せなかった標識の例。左が原画像で、右が赤でラベル付けされた画像。

表 2　標識毎の識別率 (赤－青フィルタを用いた場合)

	個別画像数	個別正解数	個別認識率	標識系列数	系列正解数	系列認識率
追い越し禁止	217	185	85.3	47	38	80.9
駐車禁止	442	400	90.5	84	83	98.8
駐停車禁止	12	11	91.7	3	4	75.0
速度制限 30Km	27	25	92.6	6	6	100
速度制限 40Km	129	101	78.3	26	22	84.6
速度制限 50Km	148	130	87.8	33	29	87.9
合計	975	852	87.4	200	191	90.5

(a) (b) (c)

図 15 赤－緑反対色フィルタと赤－青フィルタの比較。(a) 赤－青フィルタで識別できなかった標識の例。(b) 赤－緑反対色フィルタによる 2 値化。(c) 赤－青フィルタによる 2 値化。

表 3 矢印と周辺部の RGB 成分。各データは 10 点の平均値である。

	R	G	B
右矢印上部 (図)	68	128	191
右矢印上部周辺部 (地)	200	221	226
右矢印下部 (図)	150	178	200
右矢印下部周辺部 (地)	193	203	212

赤－緑反対色フィルタは、矢印の部分の R と周辺の G 成分の差を、その周辺部では周辺の R と矢印部分の G 成分の差を計算している。したがって、矢印の部分に中心があるフィルタの出力と、周辺の部分に中心があるフィルタの出力の差が比較的大きくなる。それに対して、B 成分を用いたフィルタでは、矢印と周辺部のフィルタ出力の差があまり大きくならない。2 値化のしきい値は判別分析法を用いて決定しているので、出力分布に広がりがある方がよい結果を生んでいるといえる。

青－黄反対色フィルタの場合は、赤－緑と赤－青フィルタの悪いところを集めたような結果となった。

我々の視覚系に存在しているのは、赤－緑、青－黄反対色フィルタと白－黒フィルタの 3 種であることを考えると、白地に青の配色は、我々の視覚系の機能に適合しているといえる。

6. ま と め

道路交通標識を実時間で認識するためのシステムを開発している。本報告では、赤い円形の規制標識 6 種を認識するシステムを紹介した。その特長は、HSV 表色系を用いた高速な標識候補の切り出し、赤－緑反対色フィルタを用いた標識シンボルの切り出しと構造的特徴を用いた認識にある。実験を通して、赤－緑反対色フィルタの有効性を検証することができた。また、白地に青の標識が我々の視覚系の機能に適合しているという示唆を得ることができた。

今後、構造的特徴を用いた認識系の改良と、他の規制標識や指示標識、警戒標識などの認識システムへ発展させていく予定である。

謝辞

日頃ご討論をいただいている本学福井和広助教授に感謝する。

文 献

[1] “平成 15 年中の交通事故の発生状況 (平成 16 年 2 月 26 日)”, http://www.npa.go.jp/toukei/koutuu14/h15jiko.pdf (2004).

[2] “国土交通省 ITS ホームページ”, http://www.mlit.go.jp/road/ITS/j-html/.

[3] “技術研究組合 走行支援道路システム開発機構”, http://www.ahsra.or.jp/.

[4] G. Piccioli, E. De Micheli, P. Parodi and M. Campani: “Robust method for road sign detection and recognition.”, Image and Vision Computing, **14**, pp. 209–223 (1996).

[5] A. de la Escalera, L. Moreno, M. A. Salichs and J. M. Armingol: “Road traffic sign detection and classification.”, IEEE Transaction on Industral Electronics, **44**, 6, pp. 848–859 (1997).

[6] D. M. Gavrila and V. Philomin: “Real-time object detection using distance transformation.”, Proceedings of the IEEE International Conference on Intelligent Vehicles., pp. 274–279 (1998).

[7] C. G. Sodini and S. J. Decker: “A 256x256 cmos brightness adaptive imaging array with column-parallel digital output.”, Proceedings of IEEE Interational Conference on Intelligent Vehicles, pp. 347–352 (1998).

[8] D. M. Gavrila: “Traffic sign recognition revisited.”, Proceedings of the 21st DAGM Symposium for Mustererkennung., Springer Verlag, pp. 86–93 (1999).

[9] Y. Aoyagi and T. Asakura: “A study on traffic sign recognition in scene image using geetic algorithms and neural networks.”, Proceedings of the 22nd IEEE International Conference on Industrial Electronics, Control and Instrumentation, pp. 1838–1843 (1996).

[10] 内村圭一, 木村英雄, 脇山慎也：“道路情景カラー画像における円形道路標識の抽出および認識”, 電子情報通信学会論文誌 A, **J81-A**, 4, pp. 546–553 (1998).

[11] 井上泰夫, 石川真人, 中島真人：“道路標識の自動認識”, 電子情報通信学会技術研究報告, **ITS2001-44**, 2002 年 1 月, pp. 67–72 (2002).

[12] A. de la Escalera, J. M. Armingol and M. Mata: “Traffic sign recognition and analysis for intelligent vehicles.”, Image and Vision Computing, **11**, 3, pp. 247–258 (2003).

[13] 莫, 青木由直：“カラー画像における道路標識の認識”, 電子情報通信学会論文誌 D-II, **J87-D-II**, 12, pp. 2124–2135 (2004).

[14] B. Johansson: “Road sign recognition from a moving vehicle.”, Center for Image Analysis, Swedish University of Agricultural Science and Uppsala University (2002).

[15] 原井孝輔, 山内, 高橋浩光, 松浦大祐：“認識処理に向けた円形道路標識の高精度な抽出法”, 電子情報通信学会技術研究報告, **ITS2001-44**, 2002 年 1 月, pp. 61–66 (2002).

[16] G. Loy and N. Barnes: “Fast shape-based road sign detection for a driver assistance system.”, Proceedings of IEEE/RSJ Interational Conference on Intelligent Robots and Systems (IROS2004), pp. 70–75 (2004).

[17] W. Wu, X. Chen and J. Yang: “Incremental detection of text on road signs from video with application to a driving assistant system.”, Proceedings of ACM Multimedia 2004, pp. 852–859 (2004).

[18] 内村圭一, 脇山慎也：“限定色表示を用いた円形道路標識の抽出”, 電子情報通信学会論文誌 D-II, **J83-D-II**, 2, pp. 855–858 (2000).

[19] A. de la Escalera, J. M. Armingol, J. M. Pastor and F. J. Rodoríguez: “Visual sign information extraction and identification by deformable models for intelligent vehicles.”, IEEE Transaction on Intelligent Transportation Systems, **5**, 2, pp. 57–68 (2004).

[20] J. D. Foley, A. van Dam, S. Feiner and J. F. Hughes: “Computer Graphics: Principles and Practice 2nd ed. in C”, Adisson Wesley, Reading, MA (1996). (邦訳：「コンピュータグラフィクス理論と実践」佐藤義雄監訳、オーム社、2001 年).

[21] 岩井儀雄, 松村朱里, 伊藤陽生, 清水敏之, 不殿健治, 横地裕次, 山崎隆一, 山本隆史, 村山健二, 池田：“2001 年 prmu アルゴリズムコンテスト「道路交通標識の認識」実施報告とその入賞アルゴリズムの紹介”, 電子情報通信学会技術研究報告, **PRMU2001-193**, 2001 年 12 月, pp. 125–132 (2001).

[22] 森, 坂倉栂子：“画像認識の基礎〔I〕－前処理と形の特徴抽出－”, オーム社 (1986).

反対色フィルタによる色画像のセグメンテーション
－道路交通標識の実時間認識への応用－
Image Segmentation by an Opponent-color Filter
－ Real-time Recognition of Road Sign Symbols －

馬場 今日子 (PY)†, 平井 有三‡
Kyoko Baba(PY) and Yuzo Hirai
† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
baba@viplab.is.tsukuba.ac.jp

Abstract— Opponent-color filters found in mammalian visual systems are applied to an image segmentation task as a part of real-time recognition of road sign symbols. The system consists of (1) detection of red annular objects, (2) segmentation of sign simbols, and (3) identification of symbols. Red-green opponent-color filters are used for the segmentation of symbol parts. The performance of the system has been evaluated by using about 2-hour video images taken in the daytime. Recognition rate was more than 97%. It is also suggested that the colors used in road signs are mathced to our visual systems.

Keywords—**ITS, road traffic signs, opponent-color filter, HSV color space**

1 はじめに

ニューラルネットワークの工学的応用は，BP や SVM など識別関数の学習アルゴリズムが盛んである．しかしながら，パターン認識では特徴抽出も重要な課題であり，視覚系にみられる多重解像度やガボールフィルタなど広く工学的に応用されてきた．本報告では，ほ乳類の網膜出力細胞にみられる反対色型フィルタが，色画像のセグメンテーションに効果的に利用できることを，道路交通標識の実時間認識を通して示す．

平成 15 年に国内で発生した交通事故件数は約 95 万件で，7,700 人が死亡し，118 万人が負傷している [1]．この様な背景の下，世界で ITS(Intelligent Transport System) に関する研究が行われ，国内ではその一環として，自動車運転の安全と円滑さを目標に走行支援道路システム (AHS:Advanced cruise-assist Highway System) に関する研究が多くの研究機関で行われている [2]．道路交通標識の認識に関する最近の研究例には [3] などがある．

2 HSV 表色系を用いた標識部のラベル付けと円形領域の切り出し

道路交通標識の中で規制標識の配色は，赤，白，青の組み合わせで構成されている．したがって，色を手がかりにすれば，道路画像から画素数に比例した時間で標識候補部分を切り出すことが可能になる．本論文では，赤い円形の標識を識別対象とした．色相を用いて標識候補を切り出すため，RGB 表色系から HSV 表色系へ変換し [4]，赤い対象を標識候補としてラベル付けした．

赤い円形の標識を見つけるため，「赤」にラベル付けされた画像の中から輪郭が円形の対象を検出し，標識候補とする．円形であるか否かの判定は，対象のエッジをトレースした輪郭線上の 3 点から円の方程式を求め，輪郭線と求めた円との誤差を評価して決めた．

道路交通標識は，木や電信柱により部分的に隠蔽されている場合もある．また，ビデオカメラの色のにじみなどの原因により二つの標識がつながってしまう場合がある．このような場合でも切り出せるように，円の方程式を当てはめるべき輪郭線の部分系列を同定し，不完全な円形領域も標識候補として切り出せるようにした．このような処理で，切り出し率を 5%以上上げることができた．

3 標識内部の記号の切り出しと認識

切り出された道路交通標識を識別するためには，標識内部の記号を識別する必要がある．本研究では，切り出された記号部分の構造的特徴を用いて認識を行った．記号部分の切り出しには，Laplacian-Gaussian 型の赤－緑反対色フィルタを用いた．

3.1 赤－緑反対色フィルタと判別分析 2 値化法を用いたシンボル抽出

赤－緑反対色フィルタは受容野の中心に赤色の光刺激が与えられると細胞は反応し，周辺部に緑の光刺激が与えられると反応が抑制される．また，赤と緑を含む白色光や黄色の光が受容や全体に与えられると，興奮と抑制がキャンセルしあって反応しない．また，どちらの光も含まない青に対しても反応しない．図 1 に反対色フィルタをかけた例を示した．図 A が原画像であり，図 B が赤－緑反対色フィルタの出力である．赤とそれ以外の部分を区別するために，判別分析 2 値化法を用いて 2 値化した．判別分析 2 値化法は，画像の濃度ヒストグラ

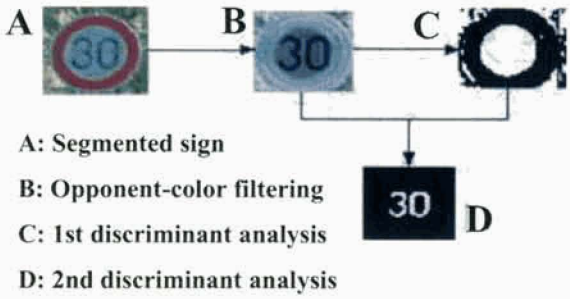


図 1: 反対色フィルタと判別分析 2 値化法を用いたシンボル部の抽出.

ムに判別分析法を適用し，濃度の明るいクラスと暗いクラスの識別境界を求め，画像を 2 値化する手法である．図 B に対して判別分析二値化法を行った結果が図 C である．さらに，青と背景の白を分離するため，図 C の中心部分に対してもう一度判別分析 2 値化法を適用した．その結果を図 D に示した．

3.2 道路交通標識の認識

道路交通標識の認識は，標識内部のシンボル部の構造的特徴を抽出し，決定木によって識別を行った．この過程を図 2 に示した．駐停車禁止は，反対色フィルターによって抽出されたシンボル部の上下の対称性の有無によって，駐車禁止は，斜め方向の軸対称性によって識別している．追い越し禁止は垂直射影した画素数の分布に見られる中央部のくびれを用いて識別している．速度規制の数字は，垂直射影に二つの分離した分布があることを手がかりとして類別し，3，4，5 の識別は交差特徴抽出法として知られている簡便な構造的特徴を用いて行っている．いずれの標識にも分類されなかった場合は，リジェクトとしている．

4 実映像による性能評価

以上のシステムを，VisualStudio.Net 2003 上で C++ を用いて実装した．ここでは，ビデオカメラ (Panasonic AG-DVX100A，レンズの焦点距離は 35mm 換算で 32.5mm) により市街地を昼間約 2 時間撮影した映像を，PC(Dell Precision 340, 2.8GHz, 2GB メモリ) に一旦取り込み，ビデオレートで再生しながら解析した結果を示す．画像サイズは 720x480 画素で，1 フレーム中に 3 個の標識がある場合の切り出しに要する時間は約 83 ミリ秒，それらの認識に要する時間は約 37 ミリ秒であった．

4.1 実験結果

約 2 時間の映像中に存在した標識の総数は 212 個で，そのうちの 200 個の標識を標識候補として切り出すことができた (切り出し率 94.3%)．200 個の標識系列には，975 枚（フレーム）の個別画像が含まれていた．そのうち，912 枚を正しく識別することができた (個別認識率：93.5%)．また，200 個の標識系列のうち，195 個の系列

図 2: 識別決定木．速度規制の構造的特徴は，30Km のみを示している．

で最後の画像を正しく識別することができた (系列認識率：97.5%)．また，標識以外の対象を標識候補として切り出した例は 14 あった．

5 まとめ

本研究で対象にした標識は赤と青を含んでいるので，赤－青フィルタの方が工学的には最適である可能性がある．しかしながら，赤－緑反対色フィルタの方が認識率がよかった．この結果は，標識の配色が我々の視覚系の特性と合致していることを示唆しているものと考えられる．

今後，構造的特徴を用いた認識系の改良と，黄－青反対色フィルタを用いた他の規制標識や指示標識，警戒標識などの認識システムへ発展させていく予定である．

参考文献

[1] “平成 15 年中の交通事故の発生状況 (平成 16 年 2 月 26 日)”, http://www.npa.go.jp/toukei/koutuu14/h15jiko.pdf (2004).

[2] “国土交通省 ITS ホームページ”, http://www.mlit.go.jp/road/ITS/j-html/.

[3] 莫, 青木由直：“カラー画像における道路標識の認識”, 電子情報通信学会論文誌 D-II, **J87-D-II**, 12, pp. 2124–2135 (2004).

[4] J. D. Foley, A. van Dam, S. Feiner and J. F. Hughes: “Computer Graphics: Principles and Practice 2nd ed. in C”, Adisson Wesley, Reading, MA (1996). (邦訳：「コンピュータグラフィクス理論と実践」佐藤義雄監訳、オーム社、2001 年).

pp.325-330

反対色型視覚フィルタによる道路交通標識の実時間認識

馬場今日子† 平井有三††

† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士前期課程
つくば市天王台 1-1-1
†† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
つくば市天王台 1-1-1
E-mail: †baba@viplab.is.tsukuba.ac.jp, ††hirai@cs.tsukuba.ac.jp

あらまし 安全運転支援を目的に、車載カラービデオカメラを用いた実時間道路交通標識認識システムを開発している。本報告では、赤い円環状の輪郭を持った標識 6 種を対象として構成した実験系について報告する。認識系は、(1)HSV 色空間と円の方程式の当てはめによる赤い円形の対象の検出処理、(2) 反対色フィルタと判別分析法による円内部のシンボル部の切り出しと、(3) 識別処理からなる。シンボル部の切り出しには、人の視覚系にある赤ー緑反対色フィルタを使用した。約 2 時間撮影した道路映像を用いて性能を評価した。その結果、97%を超える認識率が得られた。また、標識の配色が我々の視覚系と適合しているという示唆を得ることができた。

A Real-time Recognition System of Road Sign Symbols using a Visual Opponent-color Filter.

Kyoko BABA† and Yuzo HIRAI††

† Graduate School of Systems and Information Engineering, Major of Computer Science, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
†† Graduate School of Systems and Information Engineering, Deartment of Computer Science, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
E-mail: †baba@viplab.is.tsukuba.ac.jp, ††hirai@cs.tsukuba.ac.jp

Abstract In order to support safe driving, a real-time recognition system for road traffic signs has been developing. The system consists of (1) detection of red annular objects, and (2) segmentation of sign simbols, and (3) identification of symbols. Red-green opponent-color filters found in our visual systems are used for the segmentation of symbol parts. The performance of the system has been evaluated by using about 2-hour video images taken in the daytime. Recognition rate was more than 97%. Besides, it is also suggested that the combinations of colors of road signs may be adequate to our visual systems.

Key words ITS, safe-driving support, road traffic signs, opponent-color filter, HSV color space

1. はじめに

平成 15 年に国内で発生した交通事故件数は約 95 万件で、7,700 人が死亡し、118 万人が負傷している [1]。発生件数の約 6 割を追突事故と出会いがしらの事故が占めている。死亡者数は昭和 45 年の 16,000 人をピークに減少傾向にあるが、発生件数は車の保有台数に比例して増加し続けている。米国でも、2002 年に 680 万件の事故で 42,000 人が亡くなっている。この様な背景の下、世界で ITS(Intelligent Transport System) に関する研究が 1990 年頃から盛んに行われてきた。国内では高度道路交通システム (ITS) [2] の一環として、自動車運転の安全と円滑さを目標に走行支援道路システム (AHS:Advanced cruise-assist Highway System) [3] に関する研究が多くの研究機関で行われている。

AHS には AHS-i、AHS-c、AHS-a の三つのレベルがあるとされる。AHS-i は情報収集の一部をシステムがサポートするものであり、前方障害物情報の表示板による提供など現在実用化が最も進んでいるレベルである。AHS-c は運転操作の一部をシ

ステムがサポートするレベルであり、AHS-a は完全自動運転のレベルである。

AHS-i における情報収集にも色々な対象が考えられる。現在実用化されている前方障害物情報は数 km のレンジであり、運転者からは見えない情報である。しかしながら、平成 15 年の事故統計 [1] によれば、安全不確認、脇見運転、動静不注視などの事故原因が全体の 55%を占めている。したがって、運転者から見える情報を見落とさないように支援することも重要な課題である。

本研究の目的は、運転者から見える情報の見落とし防止を支援する技術を開発することにある。道路交通標識は、運転者にとって最も身近な見える対象であり、かつその見落としが直接大きな事故につながる危険性がある。道路交通標識を車載ビデオカメラを用いて認識しようとする試みは、1990 年代から海外 [4] [5] [6] [7] [8] や国内 [9] [10] で活発に行われている。

ビデオカメラを用いた道路交通標識認識は、どの研究でもおおよそ以下の手順に従って行われている。

（1） 色情報を用いた標識部の識別
（2） 幾何学的情報を用いた標識部の切り出し
（3） 標識内部のシンボル部分の切り出し
（4） シンボル部分の識別

色による標識部の識別には、RGB 表色系から色相と彩度と明度による HSV 表色系へ変換し、主として色相を手がかりに識別を行うものがある [11] [12] [13]。しかしながら、HSV 表色系は彩度が低いところで色相が不安定になることから [14]、RGB 減算法を用いたもの [15] や、RGB 間の比を用いたもの [5] が提案されている。また、色相は照明条件や距離や年月によって変化することから、濃淡画像のみを使用するもの [9] [6] [16] などが提案されている。

幾何学的情報を用いた標識部の切り出しには、エッジの勾配方向へ引いた直線の交点から円の中心を求めるもの [11]、距離変換を用いたもの [6]、図形の対称性を用いたもの [17] [13] などがある。また、色と形の情報を用いてエネルギーを定義し、GA(Genetic Algorithm) や SA(Simulated Annealing) により最適化問題を解いて交通標識を検出するものなどが提案されている [9] [18] [19]。

標識内部のシンボル部の切り出しには、エッジを用いたもの [11]、背景とシンボル部の輝度差を利用したもの [13] などが提案されている。また、シンボル部の識別には、正規化相互相関を用いたもの [11]、誤差逆伝搬型ニューラルネットワークを用いたもの [19]、水平・垂直形状特徴量を用いたもの [13] などがある。

本研究では、HSV 表色系を用いてシーン中の赤い円状の標識候補をラベル付けしている。実際の認識対象は、赤い円環状の輪郭を持った標識であるが、円と円環の識別は標識内部の記号部分を抽出するときに行っている。幾何学的情報による標識部の切り出しは、ラベル付けされた領域の輪郭線に円の方程式を当てはめ、その誤差がしきい値以内の対象を標識候補とした。また、標識内部のシンボル部の切り出しは、人の視覚系にみられる赤－緑反対色フィルタと判別分析法を組み合わせて行った。

図 1　本報告で対象とした道路交通標識。

シンボルの識別は、構造的な特徴の抽出と決定木により行った。システムを PC 上に実装し、性能評価を行った。図 1 に本論文で対象とした道路交通標識を示した。

本論文の構成は以下の通りである。第 2 節で色情報を用いた標識部の識別法と、円の方程式の当てはめによる標識候補の切り出しについて述べる。第 3 節で反対色フィルタを用いたシンボル部の切り出しについて述べる。第 4 節で識別処理について述べる。第 5 節で実験結果を紹介し検討を加える。第 6 節で本論文をまとめる。

2. 道路標識候補の切り出し

道路交通標識の中で規制標識の配色は、赤、白、青の組み合わせで構成されている。したがって、色を手がかりにすれば、全画素の色情報を判断するだけで（全画素数に比例した時間で）標識候補部分を切り出すことが可能になる。本研究では、他の多くの研究でも利用されている HSV 色空間を用いて標識部を識別する。

2.1 HSV 表色系を用いた標識部のラベル付け

色相 (Hue)、彩度 (Saturation) と明度 (Value または Brightness) は、物体表面色を規定したマンセル表色系の色を表す次元である。多くの場合、RGB 表色系から物体の色を直接想像することが困難であるのに対し、HSV 表色系は人間の感覚にあった色表現を可能にする。RGB 表色系から HSV 表色系への変換は、RGB 表色系におけるそれぞれの要素の値を、R、G、B と表現し、HSV 表色系における色相を H、彩度を S、明度を V と表現すれば以下のようになる [20]。

$$MAX = \max\{R,G,B\}, \quad MIN = \min\{R,G,B\}$$

$$H = \begin{cases} \left(0 + \frac{G-B}{MAX-MIN}\right) \times 60 & \text{if } R = MAX \\ \left(2 + \frac{B-R}{MAX-MIN}\right) \times 60 & \text{if } G = MAX \\ \left(4 + \frac{R-G}{MAX-MIN}\right) \times 60 & \text{if } B = MAX \end{cases} \quad (1)$$

$$S = \frac{MAX - MIN}{MAX} \quad (2)$$

$$V = MAX \quad (3)$$

H は 0 から 360 度まで変化し、360 度 = 0 度とする。各 R、G、B の値は、0 から 1 の間に正規化されているものとする。本研究では、赤い円状の標識を検出対象としているので、

$$324 \leqq H \leqq 28.8, \quad S \geqq 0.2, \quad V \geqq 0.1 \quad (4)$$

となる画素を、標識の赤い部分として「赤」にラベル付けする。実際の認識対象は、赤い円環状の輪郭を持った標識であるが、

(a) (b)

図 2 (a) 赤でラベル付けされた標識候補画像。分かりやすくするために、文字部の青もラベル付けしている。(b) 実際の輪郭 (内側の線) と当てはめた円。

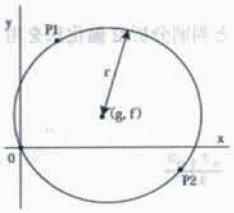


図 3 円の方程式を求める 3 点アルゴリズム

円と円環の識別は標識内部の記号部分を抽出するときに行っている。彩度が低い部分の色相は不安定になるので、判断していない [14]。画像のサイズは 720 × 480 画素である。

2.2 円形領域の切り出し

本論文で対象としている標識は赤い円環状の輪郭を持った標識なので、前節で「赤」にラベル付けされた画像の中から輪郭が円形の対象を検出し、標識候補とする。円形であるか否かの判定は、対象のエッジをトレースした輪郭線上の 3 点から円の方程式を求め、輪郭線と求めた円との誤差を評価して決める。広く利用されているハフ変換と比較して遥かに高速な処理が可能となっている。図 3 に輪郭線上の 3 点を用いて円の方程式を求める様子を示した。3 点のうち 1 点を座標原点上にとれば、円の方程式は

$$x^2 + y^2 - 2gx - 2fy = 0 \tag{5}$$

となる。(g, f) は円の中心座標であり、求める円のパラメータである。残りの 2 点 $p_1 = (x_1, y_1)$ と $p_2 = (x_2, y_2)$ から、円のパラメータは

$$\begin{cases} x_1^2 + y_1^2 - 2gx_1 - 2fy_1 = 0 \\ x_2^2 + y_2^2 - 2gx_2 - 2fy_2 = 0 \end{cases} \tag{6}$$

を解くことで求めることができる。円の半径 r は、$r = \sqrt{g^2 + f^2}$ である。

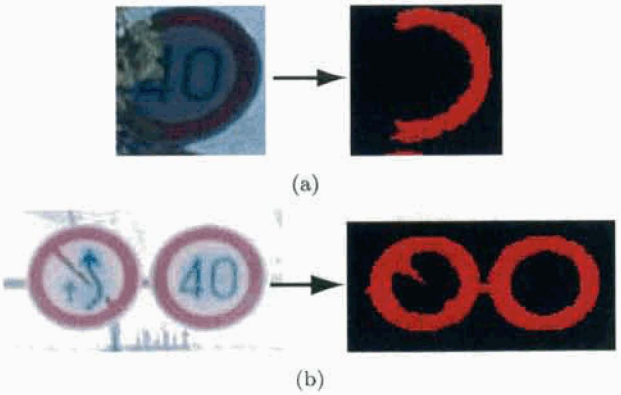
求めた円の方程式と輪郭線上のすべての点との誤差を評価し、円であるか否かの判定を行った。輪郭線上の i 番目の点と円の中心までの距離を q_i、点の総数を N とすれば、誤差 E は

$$E = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{r}{q_i}\right)^2 \tag{7}$$

で評価する。実験により、$E \leqq 0.015$ の場合、円と判断した。図 2 に当てはめた例を示した。

2.3 円の補完と分節

道路交通標識は、見通しの良いところだけに立てられているわけではなく、木や電信柱により部分的に隠蔽されている場合



(a)

(b)

図 4 (a) 木によって一部が隠された標識 (左) と赤でラベル付けした結果 (右)。(b) ビデオカメラの色のにじみにより、二つの標識 (左) が赤でラベル付けするとつながってしまった例 (右)。

(a) (b) (c)

図 5 (a) 完全な円の状態系列。(b) 図 4(a) の場合、部分系列 1 → 2 → 3 があるので、円の方程式を当てはめる。(c) 図 4(b) の場合、円の系列 3 → 0 → 1 → 2 がある。

もある。図 4(a) は木により一部が隠されている例であり、「赤」でラベル付けした結果が右の図である。また、ビデオカメラの色処理も不完全であり、にじみなどの原因により図 (b) のように二つの標識がつながってしまう場合がある。認識精度を上げるためには、このような状況下でも、図 (a) のような場合は一つの標識、図 (b) のような場合は二つの標識候補として切り出せることが必要である。そのために、円の方程式を当てはめるべき輪郭線の部分系列を同定することが必要となる。

円の方程式を当てはめてよい輪郭線の部分系列の同定は、エッジを時計回りにトレースするときの 5 点間隔の座標値の増減により状態を定義し、その状態系列で判断する。すなわち、エッジ点列の状態を下記のように定義する。ただし、座標 x、y は通常のデカルト座標系に合わせるものとする。

座標値の増減	$x \downarrow y \uparrow$	$x \uparrow y \uparrow$	$x \uparrow y \downarrow$	$x \downarrow y \downarrow$
状態	0	1	2	3

図 5(a) に示したように、通常の円の場合、状態系列は 0 → 1 → 2 → 3 のように推移する。この場合、状態 3 と状態 0 はつながっているものと考え、系列の開始位置はどこにあってもよいものとする。図 4(a) のように円が不完全な場合に対応するため、円を構成する 4 つの状態の系列のうち、3 つ以上の状態が連続して現れる場合には、そのエッジ点列に対して円の方程式を当てはめてよいと判断する。したがって、図 5(b) に示されているように、隠蔽によって一部が欠けた円の場合にも標識候補として切り出すことができる。この場合、欠けた円環の内側の輪郭もトレースされるが、図に示されているように円としての部分系列 1 → 2 → 3 → 2 が含まれているので、円と

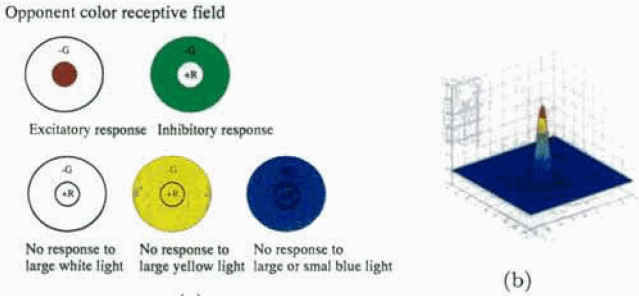


図 6 (a) 赤－緑反対色受容野を持つ神経節細胞の色刺激に対する反応特性。受容野の中心に赤い光が当たると興奮し (上左)、周辺に緑の光が当たると活動が抑制される (上右)。白色光や黄色、青の光が受容や全体に当たると興奮と抑制がキャンセルして反応しない (下左、下中、下右)。(b) 反対色フィルタの Laplacian Gaussian 関数モデル。

判断されている。また、図 (c) のように、二つの標識がつながってしまっている場合も、円を構成する状態系列 $3 \rightarrow 0 \rightarrow 1 \rightarrow 2$ があるので、それぞれを一つの標識候補として切り出すことができる。このような処理で、切り出し率を 5%以上上げることができた。

3. 標識内部の記号の切り出し

切り出された道路交通標識を識別するためには、標識内部の記号を識別する必要がある。内部の記号を切り出す処理を行わず、標識全体の大きさを正規化し、形が似たすべての鋳型との画像ベースの正規化相互相関により識別する方法も提案されているが [19]、車の移動により標識の大きさは常に変化するので大きさの正規化が不要な認識手法の方が望ましい。また、標識内部の記号の形やそれらの相対的位置は、地域ごとに異なっており、これらの変動を吸収できる認識手法の方が望ましい。したがって、本研究では標識内部の記号部分を切り出し、構造的特徴を用いた認識手法を用いた。

記号部分の切り出しには、近傍間での色差を検出すると同時にノイズ除去機能も有する、反対色フィルタによる切り出し手法を提案する。

3.1 反対色フィルタ

反対色フィルタは、ほ乳類の網膜神経節細胞や外側膝状体と呼ばれる中継核にみられる反対色細胞に相当する。代表的な反対色細胞の一つである赤－緑反対色細胞の受容野の例と色光に対する反応特性を図 6(a) に示した。図に示したように、受容野の中心に赤色の光刺激が与えられると細胞は反応し、周辺部に緑の光刺激が与えられると反応が抑制される。(このような細胞を+R-G と略記する。) また、赤と緑を含む白色光や黄色の光が受容や全体に与えられると、興奮と抑制がキャンセルしあって反応しない。また、どちらの光も含まない青に対しても反応しない。赤－緑反対色細胞には、+R-G 型以外に、-R+G、+G-R、-G+R 型がある。赤－緑型以外には、青－黄型、白－黒型がある。

このような受容野の空間的な広がりを表すモデルとして、ガウス関数を 2 回微分した Laplacian-Gaussian 関数がよく使わ

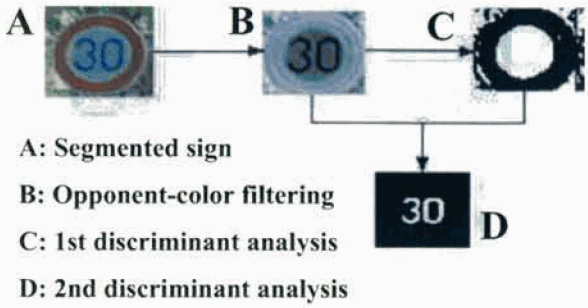


図 7 反対色フィルタと判別分析 2 値化法を用いたシンボル部の抽出。

れる。すなわち、

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{8}$$

$$F(x,y) = -\nabla^2 G_\sigma(x,y) = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{9}$$

$$F^+(x,y) = \begin{cases} F(x,y) & \text{if } F(x,y) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

$$F^-(x,y) = \begin{cases} 0 & \text{if } F(x,y) > 0, \\ F(x,y) & \text{otherwise.} \end{cases} \tag{11}$$

で定義される。関数の外形を図 6(b) に示した。

RGB 表色系で表された入力画像の位置 (x,y) における各色要素を $R(x,y)$、$G(x,y)$、$B(x,y)$ とすれば、+R-G 型の反対色フィルタの出力は、

$$O(x,y) = \int\int F^+(\xi,\eta) R(\xi - x, \eta - y) d\xi d\eta + \int\int F^-(\xi,\eta) G(\xi - x, \eta - y) d\xi d\eta \tag{12}$$

と表現される。ガウスフィルタは平滑化フィルタとして知られており、赤、緑成分のノイズを除去しながら近傍間の赤－緑成分間の差を検出していると考えることができる。また、近傍が同じ色であれば、画像の赤い部分からは正の出力、緑の部分からは負の出力、白い部分からは 0 付近の出力が得られることになる。

3.2 判別分析 2 値化法によるシンボル抽出

判別分析 2 値化法は、画像の濃度ヒストグラムに判別分析法を適用し、濃度の明るいクラスと暗いクラスの識別境界を求め、画像を 2 値化する手法である。図 7 に判別分析 2 値化法を用いたシンボル抽出法を示した。切り出された標識候補 (A) に赤－緑反対色フィルタをかけた結果が B の画像である。

判別分析法は、クラスの識別境界を、クラス内分散とクラス間分散の比を最大にする濃度値とする。出力値 t を変数として評価関数を書けば、

$$f(t) = \frac{\text{クラス間分散}}{\text{クラス内分散}} = \frac{P(C_1)(\eta_1 - \bar{\eta})^2 + P(C_2)(\eta_2 - \bar{\eta})^2}{P(C_1)\sigma_1^2 + P(C_2)\sigma_2^2} \tag{13}$$

となる。ここで、$P(C_i), i = 1,2$ はクラス i の比率、η_i はクラス内平均、$\bar{\eta}$ は全平均、σ_i^2 はクラス内分散である。

評価関数が最大値を示す出力値を 2 値化のためのしきい値とすればよい。B 図の場合、C 図に示されているように、赤い部分と、白や青の部分に分けられることになる。図 7 の C 図に 2 値化した結果を、赤側を黒、白側を白で示した。白い背景から

図 8　識別決定木。速度規制の構造的特徴は、30km/h のみを示している。詳しくは本文を参照のこと。

青い文字部を抽出するために、C の中心部分に対して、もう一度判別分析 2 値化法を適用した結果を、D 図に示す。

4. 道路交通標識の認識

道路交通標識の認識は、標識内部のシンボル部の構造的特徴を抽出し、決定木によって識別することにより行う。この過程を図 8 に示した。駐停車禁止は、反対色フィルターによって抽出されたシンボル部の対称性の有無によって、駐車禁止は、斜め方向のラベルの特徴によって識別している。追い越し禁止は垂直射影した画素数の分布に見られる中央部のくびれを用いて識別している。速度規制の数字は、垂直射影に二つの分離した分布があることを手がかりとして類別し、3、4、5 の識別は交差特徴抽出法 [21] として知られている簡便な構造的特徴を用いて行っている。いずれの標識にも分類されなかった場合は、リジェクトとしている。

5. 実映像による性能評価

以上のシステムを、VisualStudio.Net 2003 上で C++を用いて実装した。赤い物体検出や円検出などの処理モジュールを dll として作成し、(a) 各処理段階におけるパラメータ調整などを同じ画面上で行えるようにした実験システム、(b) 切り出された標識候補画像を読み込んで認識率などの性能評価を行うための評価システム、(c) 実際に車に搭載して実時間で道路交通標識を認識するシステムを開発した。ここでは、ビデオカメラ (Panasonic AG-DVX100A、レンズの焦点距離は 35mm 換算で 32.5mm) により市街地を昼間約 2 時間撮影した映像を、PC(Dell Precision 340, 2.8GHz, 2GB メモリ) に一旦取り込み、ビデオレートで再生しながら解析した結果を示す。

1 フレーム中に 3 個の標識がある場合の切り出しに要する時間は約 83 ミリ秒、それらの認識に要する時間は約 37 ミリ秒であった。標識が 1 つの場合、全体の処理時間は約 70 ミリ秒であった。したがって、平均すれば約 10 画像/秒程度で映像がサンプリングできていることになる。

5.1 実験結果

図 9 に切り出された標識画像系列と識別結果の例を示した。速度制限 40km/h の場合は最初の 2 枚と 5 枚目で識別誤りが生じている。一般的な傾向として、図 9 の左側の画像のように遠くて小さな画像の場合に誤る確率が高くなっている。また、系列の最後の画像は距離が一番近いところにあるので大きく明確な画像が得られ認識率も高くなる。評価法としては、個々に切り出された画像 (例えば図 9 の 7 枚の画像) の正解数を用いた評価と、各系列の最後の画像 (例えば図 9 の系列の最後の画像) の正解数を用いた。後者は、対象となっている標識が正しく識別できたか否かの指標として用いた。

図 9　サンプリングされた切り出された標識画像と認識結果の例。速度制限 40km/h の例。7 枚の画像を切り出すことができている。各画像の下には、識別結果が示されている。NOT:No Overtaking、S40:速度制限 40km/h。

表 1　標識毎の識別率

	個別画像数	個別正解数	個別認識率	標識系列数	系列正解数	系列認識率
追い越し禁止	217	210	96.8	47	47	100
駐車禁止	442	439	99.3	84	84	100
駐停車禁止	12	12	100	4	4	100
速度制限 30km/h	27	24	88.9	6	6	100
速度制限 40km/h	129	102	79.1	26	24	92.3
速度制限 50km/h	148	125	84.5	33	30	90.9
合計	975	912	93.5	200	195	97.5

約 2 時間の映像中に存在した標識の総数は 212 個で、そのうちの 200 個の標識を標識候補として切り出すことができた (切り出し率 94.3%)。200 個の標識系列には、975 枚の個別画像が含まれていた。そのうち、912 枚を正しく識別することができた (個別認識率：93.5%)。また、200 個の標識系列のうち、195 個の系列で最後の画像を正しく識別することができた (系列認識率：97.5%)。また、標識以外の対象を標識候補として切り出した例は 14 あった。正しくリジェクトできたのは 2 例のみであったが、リジェクトできなかったものの中に系列長が 1 のものが 7 例あった。標識の系列長はすべて 2 以上なので、これらは排除可能と考えている。

5.2 標識毎の認識率の比較

表 1 に標識毎の認識率を示した。個別正解数は、小さな画像から大きな画像まで混在しているので 93%程度の正解率にとどまっているが、系列最後の画像に対する正解率は 97%を超えている。速度制限 40km/h と 50km/h に対する系列認識率が他と比較して悪くなっている原因は、文字部の特徴抽出に簡便な交差特徴抽出法を用いていることにあるものと考えられる。例えば、文字「4」を構成する斜めの線は、2 値化すると途切れてしまうことがよく見られたが、交差特徴抽出法で特定の位置の交差情報を検出するときこの途切れ部分にかかると正しく識別できなくなる。

5.3 誤識別の例

誤識別は、標識を別の標識あるいは標識でないと判断したも

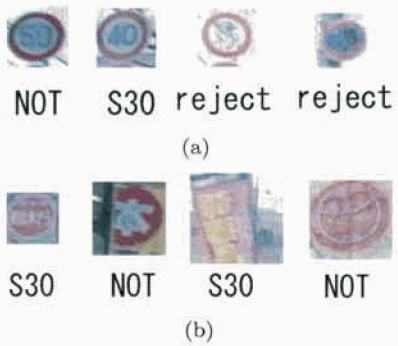


図 10　誤識別の例。(a)miss の例。いずれも系列の最後の画像。(b)false alarm の例。左の 3 つは系列長が 1 の画像。

の (miss)、標識でないものを標識と識別したもの (false alarm)、標識を切り出すことができなかったものに分類できる。これらの例を図 10 に示した。図 (a) に示した miss の画像は、いずれも系列の最後の画像である。標識の傾きや輝点ノイズが誤識別の原因となっている。傾きが原因の場合系列のすべての画像が誤識別になるので、個別識別率が低下する大きな要因となる。図 (b) は false alarm の例である。左の 3 つは系列長が 1 の画像である。正しい標識の場合、系列長は 2 以上なので、長さ 1 の系列を除外することで誤りを避けることができると考えている。一番右の画像は、15 個の画像からなる系列の最後の画像である。切り出せなかった標識の例は標識の後ろに赤い看板等がある場合である。他に、標識が小さすぎる場合にも切り出すことができていない。

5.4　フィルタによる性能の違い

視覚系には、赤-緑反対色フィルタ以外に、黄-青反対色フィルタが存在している。また、標識が赤と青で構成されていることから、赤-青フィルタの方が性能がよい可能性がある。3 種類のフィルタを比較した結果、赤-緑反対色フィルタが一番よかった。したがって、標識の配色は、我々の視覚系の特性に合っていると考えられる。

6. ま と め

道路交通標識を実時間で認識するためのシステムを開発している。本報告では、赤い円環状の輪郭を持った規制標識 6 種を認識するシステムを紹介した。その特長は、HSV 表色系を用いた高速な標識候補の切り出し、赤ー緑反対色フィルタを用いた標識シンボルの切り出しと構造的特徴を用いた認識にある。実験を通して、赤ー緑反対色フィルタの有効性を検証することができた。また、白地に青の標識が我々の視覚系の機能に適合しているという示唆を得ることができた。

今後、構造的特徴を用いた認識系の改良と、他の規制標識や指示標識、警戒標識などの認識システムへ発展させていく予定である。

謝辞

日頃ご討論をいただいている本学福井和広助教授に感謝する。

文　献

[1] “平成 15 年中の交通事故の発生状況 (平成 16 年 2 月 26 日)”, http://www.npa.go.jp/toukei/koutuu14/h15jiko.pdf (2004).

[2] “国土交通省 ITS ホームページ”, http://www.mlit.go.jp/road/ITS/j-html/.

[3] “技術研究組合 走行支援道路システム開発機構”, http://www.ahsra.or.jp/.

[4] G. Piccioli, E. De Micheli, P. Parodi and M. Campani: “Robust method for road sign detection and recognition.”, Image and Vision Computing, **14**, pp. 209–223 (1996).

[5] A. de la Escalera, L. Moreno, M. A. Salichs and J. M. Armingol: “Road traffic sign detection and classification.”, IEEE Transaction on Industral Electronics, **44**, 6, pp. 848–859 (1997).

[6] D. M. Gavrila and V. Philomin: “Real-time object detection using distance transformation.”, Proceedings of the IEEE International Conference on Intelligent Vehicles., pp. 274–279 (1998).

[7] C. G. Sodini and S. J. Decker: “A 256x256 cmos brightness adaptive imaging array with column-parallel digital output.”, Proceedings of IEEE Interational Conference on Intelligent Vehicles, pp. 347–352 (1998).

[8] D. M. Gavrila: “Traffic sign recognition revisited.”, Proceedings of the 21st DAGM Symposium for Mustererkennung., Springer Verlag, pp. 86–93 (1999).

[9] Y. Aoyagi and T. Asakura: “A study on traffic sign recognition in scene image using geetic algorithms and neural networks.”, Proceedings of the 22nd IEEE International Conference on Industrial Electronics, Control and Instrumentation, pp. 1838–1843 (1996).

[10] 内村圭一, 木村英雄, 脇山慎也：“道路情景カラー画像における円形道路標識の抽出および認識”, 電子情報通信学会論文誌 A, **J81-A**, 4, pp. 546–553 (1998).

[11] 井上泰夫, 石川真人, 中島真人：“道路標識の自動認識”, 電子情報通信学会技術研究報告, **ITS2001-44**, 2002 年 1 月, pp. 67–72 (2002).

[12] A. de la Escalera, J. M. Armingol and M. Mata: “Traffic sign recognition and analysis for intelligent vehicles.”, Image and Vision Computing, **11**, 3, pp. 247–258 (2003).

[13] 莫, 青木由直：“カラー画像における道路標識の認識”, 電子情報通信学会論文誌 D-II, **J87-D-II**, 12, pp. 2124–2135 (2004).

[14] B. Johansson: “Road sign recognition from a moving vehicle.”, Center for Image Analysis, Swedish University of Agricultural Science and Uppsala University (2002).

[15] 原井孝輔, 山内, 高橋浩光, 松浦大祐：“認識処理に向けた円形道路標識の高精度な抽出法”, 電子情報通信学会技術研究報告, **ITS2001-44**, 2002 年 1 月, pp. 61–66 (2002).

[16] G. Loy and N. Barnes: “Fast shape-based road sign detection for a driver assistance system.”, Proceedings of IEEE/RSJ Interational Conference on Intelligent Robots and Systems (IROS2004), pp. 70–75 (2004).

[17] W. Wu, X. Chen and J. Yang: “Incremental detection of text on road signs from video with application to a driving assistant system.”, Proceedings of ACM Multimedia 2004, pp. 852–859 (2004).

[18] 内村圭一, 脇山慎也：“限定色表示を用いた円形道路標識の抽出”, 電子情報通信学会論文誌 D-II, **J83-D-II**, 2, pp. 855–858 (2000).

[19] A. de la Escalera, J. M. Armingol, J. M. Pastor and F. J. Rodoríguez: “Visual sign information extraction and identification by deformable models for intelligent vehicles.”, IEEE Transaction on Intelligent Transportation Systems, **5**, 2, pp. 57–68 (2004).

[20] J. D. Foley, A. van Dam, S. Feiner and J. F. Hughes: “Computer Graphics: Principles and Practice 2nd ed. in C”, Adisson Wesley, Reading, MA (1996). (邦訳：「コンピュータグラフィクス理論と実践」佐藤義雄監訳、オーム社、2001 年).

[21] 森, 坂倉栂子：“画像認識の基礎 [I] ー前処理と形の特徴抽出ー”, オーム社 (1986).

ImageFilterPlugin 仕様書

Version1.0

作成：石塚 裕(ishizuka@zuworks.net)

1 はじめに

ImageFilterPlugin とは，Windows の DLL 形式のファイルである．DLL から適切な関数をエクスポートすることで，ImageFilter アプリケーションから利用可能である．本仕様書に従ってプラグインを作成してください。

2 関数のエクスポート

プログラムは様々な関数からなっている．DLL にも同様に様々な関数を含めることができる．しかし，これらの関数がすべて公開されてしまっては，ライブラリとして不必要な関数まで外部に公開されてしまう．そこで，外部に公開する関数を限定する必要がある．DLL から関数をエクスポートする方法は，「モジュール定義ファイル(.def)」と“関数の修飾”である．

2.1 モジュール定義ファイルの書き方

VisualStudio.NET のウィザードでモジュール定義ファイルを作成すると次のようなファイルができる．

```
LIBRARY Filter1
```

このファイルにエクスポートする関数を追加する．書式は次の通りである．

```
LIBRARY Filter1
EXPORTS
    DoFilter    @1
    GetFilterPluginInfo    @2
```

“EXPORTS”という命令の後に，“関数名 @数字”という具合である．

2.2 関数の修飾

モジュール定義ファイルの他に，関数の修飾をする必要がある．例えば，”int function(int a)”というプロトタイプの関数をエクスポートしたい場合は，

```
int __declspec(dllexport) WINAPI function( int a );
```

とする．

__declspec(dllexport)は関数を DLL からエクスポートするという命令である．FilterPlugin.h でマクロ(DLLEXPORT)として定義してある．

WINAPI は呼び出し規約である．詳しくはここでは割愛するが必要なものであるので書かなくてはならない．

（詳細は MSDN ライブラリで，DLL 等をキーワードにして調べていただきたい．）

2.3 フィルタプラグインで外部に公開，実装する必要がある関数

ImageFilter プラグインの仕様を満たすためには，以下の二つの関数を実装する必要がある．

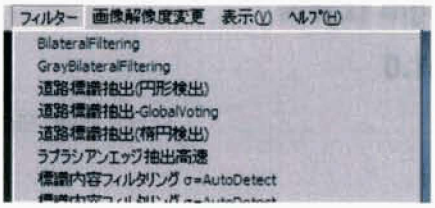
int DLLEXPORT WINAPI GetFilterPluginInfo(LPSTR buf,int buflen);

GetFilterPluginInfo - Plug-in に関する情報を得る

LPSTR buf　　：情報を納めるバッファ

int buflen：バッファ長(byte)

説明：　ImageFilter アプリケーション側からプラグインの名前を取得する時に呼ばれる．



int DLLEXPORT WINAPI DoFilter(BITMAPINFO *bi, BYTE *imagedata,PROGRESSCALLBACK lpPrgressCallback,long lData);

・DoFilter - フィルタをかける

BITMAPINFO *bi: BITMAPINFO 構造体

BYTE imagedata: 画像データ

FARPROC lpPrgressCallback :

途中経過を表示するコールバック関数へのポインタ。

コールバック関数の prototype:

int PASCAL ProgressCallback(int nNum,int nDenom,long lData);

まず nNum==0 でコールされ、nNum==nDenom になるまで

定期的に呼ばれる。

long lData : コールバック関数に渡す long データ．ポインタ等を必要に応じて受け渡せる。

ReturnValue:

1 正常終了

1 以外 不正終了

説明：フィルタ処理のメイン関数

BITMAPINFO *bi と，imagedata を利用して画像処理部分を実装してください．

BITMAPINFO *bi と BYTE imagedata は，CDIB クラスを用いると扱いが簡単になります．

＊ これらの関数の実装の仕方は，プロジェクト FilterTemplate を参考にしてください．＊

```
// FilterTemplete.cpp : DLL アプリケーション用のエントリ ポイントを定義します。
//
#include "stdafx.h"
#include "FilterPlugin.h"
#include <CDIB.h>

// DLL 用の main 関数のようなもの，基本的にこのままでよい
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                                                    )
{
    return TRUE;
}

//  プラグインの名前を返す関数－ImageFilter から呼ばれる
int DLLEXPORT WINAPI GetFilterPluginInfo(LPSTR buf,int buflen)
{
    lstrcpy( buf, "フィルターテンプレート" );
    return 1;
}
```

```
//  フィルタメイン関数
int DLLEXPORT WINAPI DoFilter( BITMAPINFO *bi, BYTE *imagedata,PROGRESSCALLBACK
lpPrgressCallback,long lData)
{
    CDIB dib;                                                    // CDIB クラスを使う．画像処理が簡
単になる．
    dib.CreateFromDirectMemoryData( bi, imagedata );     // 画像データを登録する．
    int ySize = dib.Height();                                    // 画像の高さを取得する．
    int xSize = dib.Width();                                     // 画像の幅を取得する．
    int i,j;
    BYTE r,g,b,c;

    for( i = 0; i < ySize; i++ ){
        for( j = 0; j < xSize; j++){
            b = dib.B( j,i );                                    // 座標(i,j)の色要素(B)を取得する
            g = dib.G( j,i );                                    // 座標(i,j)の色要素(G)を取得する
            r = dib.R( j,i );                                    // 座標(i,j)の色要素(R)を取得する

            dib.SetColor( j,i, RGB( 255-r, 255-g,255-b ));  //dib.SetColor でピクセルに色をつけ
る．ここではネガポジ変換をしている．

        }
    }

    return 1;
}
```

DIB，BMP 操作クラス「CDIB」仕様書

Version1.0

作成：石塚 裕(ishizuka@zuworks.net)

1　はじめに

クラス CDIB は Windows/C++用の BMP，画像操作クラスです．このクラスを用いることで煩雑な Windows の画像ファイルの操作を単純にすることができます．

Windows で BMP やそれに準じる画像を扱うときにはいくつか注意しないといけないことがあります．

1．座標の原点が左下であること．画像処理の教科書やテキストでは，たいてい左上が座標の原点になっていることが多い．

2．画像データの並びが，RGB の順ではなく，BGR の順である．これは本クラス CDIB を用いることで解決できる．

3．その画像データの並びに関して，画像の横幅が 4 の倍数でない場合．4 の倍数となるように 1 列あたりのデータを調節しなくてはならない．

このように注意しなければならないことがあることを知っておいてください．

BMP ファイルや Windows での画像の扱い方については，MSDN で" Bitmap Reference"等を検索してください．

2　簡単な使い方

簡単な使い方をいくつか示します．

BMP ファイルを作って保存する．

```
#include <stdio.h>
#include <windows.h>
#include <CDIB.h>        // CDIB クラスを使う場合は必ずインクルードする
int main()
{
    CDIB dib;
    dib.Create( 100, 100 );
    dib.DrawCircle( 50,50,10,RGB(255,0,0));
    dib.SaveToFile("SampleDIB000.bmp");
    return 0;
}
```

この例では，縦横 100x100 ピクセルの画像をつくり，(50,50)を中心に半径 10 の円を描き，SampleDIB000.bmp というファイルに保存します．

既存の BMP ファイル開いてネガポジ変換する．

```
#include <stdio.h>
#include <windows.h>
#include <CDIB.h>      // CDIB クラスを使う場合は必ずインクルードする
int main()
{
    int x,y;
    int R,G,B;
    CDIB dib;
    dib.OpenBitmap( "SampleDIB0001.bmp" );
    for( y = 0; y < dib.Height(); y++){
        for( x = 0; x < dib.Width(); x++){
            R = dib.R(x,y);
            G = dib.G(x,y);
            B = dib.B(x,y);
            dib.SetColor(x,y,RGB(255-R,255-G,255-B));
        }
    }
    dib.SaveToFile("SampleDIB0001.bmp");
```

```
    return 0;
}
```

3 詳細仕様

CDIB クラスを使う場合は，必ず，ソースファイルもしくは，ヘッダファイルで#include <CDIB.h>とすること．

コンストラクタ

コンストラクタ	説明
CDIB()	デフォルトコンストラクタ 例 CDIB dib; または CDIB *dib2 = new CDIB();
CDIB(int w, int h)	幅w，高さhの初期画像，８ビット(RGB 各色 0-255 の値) 例 CDIB dib1(320,240); CDIB *dib2 = new CDIB(320,240);
CDIB(int w, int h, int BitCount)	幅w，高さhの初期画像，ビットカウント(１ビット，４ビット，８ビットに対応) 例 CDIB dib(320,240,24); または， CDIB *dib2 = new CDIB(320,240,24);
CDIB(const char* FileName)	ファイルを開いて初期化 例 CDIB dib(“Sample.bmp”); または， CDIB *dib2 = new CDIB(“Sample.bmp”);
CDIB(CDIB &rh)	コピーコンストラクタ 例 CDIB dib(“Sample.bmp”); CDIB2 dib2 = dib; // dib2 には、”Sample.bmp”が入る．

public メンバ

返り値とメンバ関数名	説明

LONG Width()
LONG Height()
	画像の幅と高さを返す
	例
	CDIB dib(“sample.bmp”);
	int width = dib.Width(); // 幅
	int height = dib.Height(); // 高さ

WORD BitCount()	画像のビットカウント(１ピクセルあたりのビット数) 通常は 24 が返ってくる．

<table>
<tr><td>BITMAPINFO* GetBitMapInfo()</td><td>BITMAPINFO 構造体へのポインタを返す
例
CDIB dib(“sample.bmp”);
BITMAPINFO *bi = dib.GetBitMapInfo();</td></tr>
<tr><td>BITMAPINFOHEADER*
GetBitMapInfoHeader()</td><td>BITMAPINFOHEADER 構造体へのポインタを返す
例
CDIB dib(“sample.bmp”);
BITMAPINFOHEADER* bih = dib.GetBitMapInfoHeader();</td></tr>
<tr><td>BYTE* GetImageData()</td><td>画像データを返す，24 ビットカラー画像であれば，BGR の順に並んだ配列.
例
CDIB dib(“Sample.bmp”);
BYTE *imagedata = dib.GetImageData();</td></tr>
<tr><td>int GetImageSize()</td><td>画像の大きさ（バイト単位）
必ずしも，幅 x 高さと一致しない
例
CDIB dib(“Sample.bmp”);
int imagesize = dib.GetImageSize();</td></tr>
<tr><td>BYTE B(LONG idx)
BYTE B(LONG x, LONG y)
BYTE G(LONG idx)
BYTE G(LONG x, LONG y)
BYTE R(LONG idx)
BYTE R(LONG x, LONG y)</td><td>座標（x,y)、の RGB 値を返す
例
CDIB dib(“sample.bmp”)
BYTE r = dib.R(10,10);
BYTE g = dib.G(10,10);
BYTE b = dib.B(10,10);</td></tr>
<tr><td>BYTE* BGR(LONG x, LONG y)</td><td>座標（x,y)、の BGR 値の先頭を返す
BGR+1 => G
BGR+2 => R
例
CDIB dib(“sample.bmp”);
BYTE *bgrdata = dib.BGR(10,10);
// *(bgrdata) = > dib.B(10,10);と等価
// *(bgrdata+1) = > dib.G(10,10);と等価
// *(bgrdata+2) = > dib.R(10,10);と等価</td></tr>
<tr><td>BOOL IsLoad()</td><td>BMP/DIB が読み込まれていれば TRUE を返す.
双でなければ，FALSE を返す.</td></tr>
</table>

<table>
<tr><td>BYTE* ScanLine(int y)</td><td>y 行目の先頭のデータを返す.
例
CDIB dib(“sample.bmp”);
BYTE *SecondLineData = dib.ScanLine(1);</td></tr>
<tr><td>int Create(int w, int h , int BitCount = 24)</td><td>幅w，高さ h，ビットカウント 24 で画像を作成.
例
CDIB dib;
dib.Create(320,240);</td></tr>
<tr><td>int OpenBitmap(const char* FileName)</td><td>BMP ファイルを開く.
例
CDIB dib;
dib.OpenBitmap(“Sample.bmp”);</td></tr>
<tr><td>CreateFromBitMapData(
BITMAPINFO *pBmi, BYTE *pData)</td><td>BITMAOINFO 構造体と画像データ pData から CDIB を作る.
データはコピーされる.</td></tr>
<tr><td>Int CreateFromDirectMemoryData(
BITMAPINFO *pBmi, BYTE *pData)</td><td>BITMAOINFO 構造体と画像データ pData から CDIB を作る.
データはコピーされず，元の pData を操作することができる.</td></tr>
<tr><td>int DrawDIBtoHDC(HDC hdc,int x,int y)</td><td>デバイスコンテキスト(Windows で扱うキャンバスのようなもの)に表示する.</td></tr>
<tr><td>int SetColor(int x,int y,int c)</td><td>点(x,y)に色 c をつける．c は RGB()マクロを用いることができる.
例
CDIB dib(“sample.bmp”);
dib.SetColor (10,10, RGB(255,255,255));</td></tr>
<tr><td>int ImageDataCopy(CDIB &src)
int ImageDataCopy(CDIB *pDIB)</td><td>CDIB データをコピーする.
例
CDIB dib1(“Sample1.bmp”);
CDIB copyDIB;
copyDIB.ImageDataCopy(dib1);</td></tr>
<tr><td>int SaveToFile(const char *FileName)</td><td>ファイルに保存する.
例
CDIB dib(“sample.bmp”);
//何らかの処理をする.
Dib.SaveToFile(“SaveBMP.bmp”);</td></tr>
<tr><td>void ResizeBMP(
DWORD width ,DWORD height)</td><td rowspan="2">最近傍法によって画像サイズを変更する
例
CDIB dib(320,240);
dib.ResizeBMP(160,120);</td></tr>
<tr><td>void ResizeBMPbyNN(
DWORD width ,DWORD height)</td></tr>
<tr><td>void ResizeBMPbyAreaAverage(</td><td>面積平均法によって，画像サイズを縮小する.</td></tr>
</table>

int width, int height)	最近傍法よりもきれいに縮小できる.

int SelectErea(RECT *rect, CDIB *pDIB)	RECT で指定する領域を p DIB にコピーする.
	例
	CDIB dib("sample.bmp");
	CDIB selectArea;
	RECT Rect;
	//Windows の画像は左下が原点の座標系であるので top と bottom の意味が逆になることに注意
	Rect.top = 10;
	Rect.bottom = 30;
	Rect.left = 10;
	Rect.right = 40;
	dib.SelectArea(&rect, &selectArea);

int AddDIB(int x, int y, CDIB *pDIB)	任意の位置(x,y)に p DIB を上書きする.
int AlphaRender(CDIB *pSuf, double a ,int x = 0, int y=0)	任意の位置(x,y)に α 値(0~1.0)の割合で pSuf をレンダリングする.
int Splite(CDIB *pSuf, int x = 0, int y = 0 ,int c = -1)	クロマキー値 c で示す色をクロマキーにして, 任意の位置(x,y)にレンダリングする. C=-1 だった場合, (x,y)=(0,0)の色がクロマキーになる.
int ANDRender(CDIB *pSuf,int x = 0, int y=0)	各ピクセルの AND 値をとり, 任意の位置(x,y)に上書きする.
void Init()	内容を初期化する.
void MemoryFree()	メモリを解放する.
inline int DrawLine(int x0, int y0, int x1, int y1 ,int color)	(x0,y0)から(x1,y1)に色 color で直線を引く
int DrawCircle (int start_x, int start_y, int R ,int color)	中心(x,y), 半径 r, 色 color の円を描く.
int DrawSquare (int x1, int y1, int x2, int y2, int color)	左下の点(x1,y1), 右上の点(x2,y2)で囲まれる領域で色 color の四角形を描く
int PaintSquare (int x1, int y1, int x2, int y2, int color)	左下の点(x1,y1), 右上の点(x2,y2)で囲まれる領域を色 color で塗りつぶす
int DrawEllipse (int Cx, int Cy, int a, int b ,double rot, int color)	中心(Cx,Cy),軸の長さ a,b, 傾き rot, 色 color の楕円を描く.

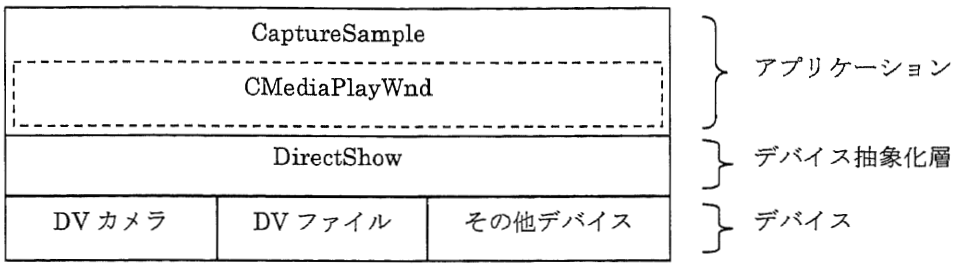
2005/03/15

CaptureSample 追加資料

2005/03/15　THasunuma

システム構成

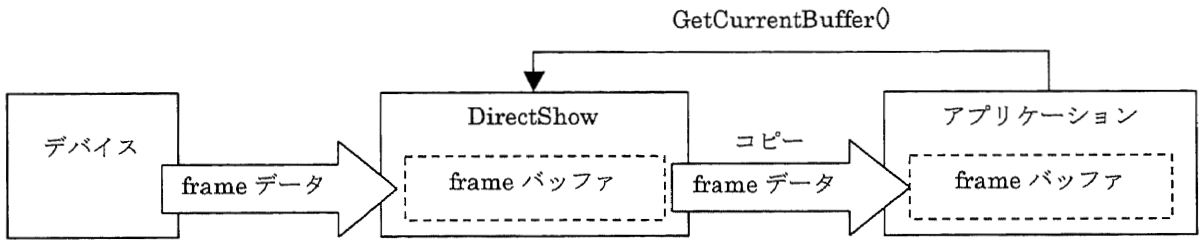
サンプルアプリケーション（CaptureSample）の構成を示す。



DV カメラ・DV ファイル等の各デバイスは、DirectShow に依って抽象化され、アプリケーション（CaptureSample）からは、CMediaPlayWnd クラスを通して操作される。

manual 動作（アプリケーション主導キャプチャ）

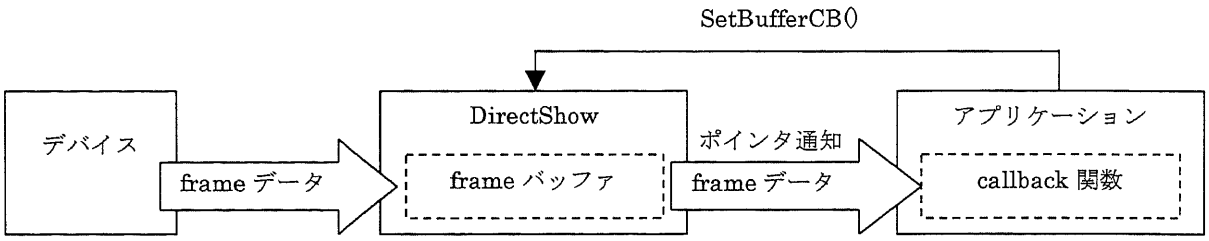
manual 動作時の概念図を示す。



DirectShow は、デバイスからのフレームデータを随時受け付ける。
そして、新しいフレームデータが到着する毎に前のフレームデータは上書きされる。
アプリケーションは、必要に応じて CMediaPlayWnd::GetCurrentBuffer()を呼び出し、最新のフレームデータを取得する。
この際、フレームデータを受け取るバッファはアプリケーション側で用意する。

callback 動作（デバイス主導キャプチャ）

callback 動作時の概念図を示す。



アプリケーションは、CMediaPlayWnd::SetBufferCB()を呼び出し、コールバック関数を登録しておく。

DirectShow は、デバイスからのフレームデータを随時受け付け、新しいフレームデータが到着する毎に登録されているコールバック関数を呼び出す。

この際、DirectShow は内部で使用しているフレームバッファのポインタを通知する。そして、callback 関数が終了する迄、デバイスから新たに送られてくるフレームデータは破棄される。（callback 関数が多重に呼び出される事はない様だ）

```
//------------------------------------------------------------------------------
//
//  CSampleGrabberCB クラス
//
//      CSampleGrabberCB.h
//
//  VC++.NET 2003               2004/11/01  THas
//
//------------------------------------------------------------------------------
#ifndef __CSampleGrabberCB_h__
#define __CSampleGrabberCB_h__
//------------------------------------------------------------------------------
//  include files
//------------------------------------------------------------------------------

#include <qedit.h>


//------------------------------------------------------------------------------
//  definitions
//------------------------------------------------------------------------------

#ifndef BUFFERCBFUNCTION
typedef int (*BUFFERCBFUNCTION) (double dTime, BYTE *pBuffer, long nBufferSize, void
*pData);
#endif


//------------------------------------------------------------------------------
//  CSampleGrabberCB class
//------------------------------------------------------------------------------

class CSampleGrabberCB : public ISampleGrabberCB
{
    protected:

        BUFFERCBFUNCTION    m_pBufferCB;
        void                *m_pData;


    public:

        //  constructor
        CSampleGrabberCB()
        {
            m_pBufferCB = NULL;
            m_pData     = NULL;
        }

        //  destructor
        virtual ~CSampleGrabberCB() {    }
```



```
        //  COM ref counting
        STDMETHODIMP_(ULONG) AddRef()  {  return 1;  }
        STDMETHODIMP_(ULONG) Release() {  return 2;  }

        //  Fake out any COM QI'ing
        STDMETHODIMP QueryInterface(REFIID riid, void **ppv)
        {
            if(NULL == ppv)  return E_POINTER;

            if(riid == __uuidof(IUnknown)) {
                *ppv = static_cast<IUnknown*>(this);
                return S_OK;
            }

            if(riid == __uuidof(ISampleGrabberCB)) {
                *ppv = static_cast<ISampleGrabberCB*>(this);
                return S_OK;
            }

            return E_NOINTERFACE;
        }

        //  ISampleGrabberCB functions
        STDMETHODIMP SampleCB(double dTime, IMediaSample *pSample)
        {
            return E_NOTIMPL;
        }

        STDMETHODIMP BufferCB(double dTime, BYTE *pBuffer, long nBufferSize)
        {
            if(NULL != m_pBufferCB) {
                return (*m_pBufferCB)(dTime, pBuffer, nBufferSize, m_pData);
            } else {
                return S_OK;
            }
        }

        //  methods
        virtual void SetBufferCBFunction(BUFFERCBFUNCTION pFunc, void *pData)
        {
            m_pBufferCB = pFunc;
            m_pData     = pData;
        }

};


//------------------------------------------------------------------------------
#endif
```

```
// ImageFilterPlgin.h: CImageFilterPlgin クラスのインターフェイス
//
//////////////////////////////////////////////////////////////////////

#if !defined(___IMAGEFILTERPLGIN_H___)
#define ___IMAGEFILTERPLGIN_H___


#include <windows.h>
#include <io.h>

#include <vector>
#include <string>
#include <algorithm>
using namespace std;

// for DEBUG
//#include <support_functions.h>

#pragma warning( disable : 4236 )
#pragma warning( disable : 4786 )

#define DLLIMPORT   __declspec(dllimport)
//int DLLIMPORT WINAPI GetFilterPluginInfo(LPSTR, int );
typedef int (WINAPI *GETFILTERPLUGININFO)(LPSTR ,int);

typedef int (WINAPI *DOFILTER)( BITMAPINFO*, BYTE*,FARPROC ,long);

class CImageFilterPlgin
{
    HINSTANCE   hDLL;
    char        _plugin_path_name[ MAX_PATH ];

    GETFILTERPLUGININFO     pGetFilterPluginInfo;
    DOFILTER            pDoFilter;

public:
    CImageFilterPlgin(){
    }
    virtual ~CImageFilterPlgin(){
        FreeLibrary( hDLL );
    }

    // プラグインローダー
    int LoadPlugin(const char* PluginName)
    {
        pGetFilterPluginInfo    =   NULL;
        pDoFilter       =   NULL;

        hDLL = ::LoadLibrary( PluginName );    // DLLのロード
        if (hDLL != NULL)
        {
            pGetFilterPluginInfo    =   (GETFILTERPLUGININFO)::GetProcAddress(hDLL,"G
```



```
etFilterPluginInfo" );
            pDoFilter   =   (DOFILTER)::GetProcAddress( hDLL, "DoFilter" );
            if( pGetFilterPluginInfo && pDoFilter ){
                strncpy( _plugin_path_name, PluginName, MAX_PATH );
                return TRUE;
            }else{
                return FALSE;
            }
        }else{
            return FALSE;
        }

    }



    // プラグイン関数
    //
    int GetFilterPluginInfo(LPSTR buf,int buflen)
    {
        if( pGetFilterPluginInfo )
            return (*pGetFilterPluginInfo)(  buf, buflen );
        else
            return 0;
    }

    int DoFilter(BITMAPINFO *bi, BYTE *imagedata,FARPROC lpPrgressCallback,long
lData)
    {
        if( pDoFilter )
            return (*pDoFilter)( bi, imagedata,lpPrgressCallback,lData );
        else
            return 0;
    }


};


class CImageFilterPlginLoader
{
    vector< CImageFilterPlgin* >        m_vPlgins;
    vector< string >                    m_vPlginNames;

public:
    CImageFilterPlginLoader(){

    }
    virtual ~CImageFilterPlginLoader(){
        FreePlugins();
    }
```

```
int SetFilterPluginDirectory( const char *path )
{
    HANDLE              fh;
    WIN32_FIND_DATA     ff;
    char    *PlginPath, *wk;
    CImageFilterPlgin   *plgin;
    vector< string >    vplgins;


    PlginPath = new char[ lstrlen( path ) + MAX_PATH ];
    wk = new char[ lstrlen(path ) + MAX_PATH ];

    sprintf( PlginPath, "%s\\*.dll", path );

    fh = FindFirstFile( PlginPath, &ff );
    while( fh != INVALID_HANDLE_VALUE ){
        sprintf( wk, "%s\\%s", path, ff.cFileName );
        vplgins.push_back( wk );
        if (FindNextFile(fh, &ff) == false) break;
    }

    //  プラグインをソートしてロード
    sort( vplgins.begin(), vplgins.end());
    vector<string>::iterator it_plgin = vplgins.begin();
    for( ; it_plgin != vplgins.end(); it_plgin++){
        plgin = new CImageFilterPlgin();
        if( plgin->LoadPlugin( it_plgin->c_str() ) ){
            m_vPlgins.push_back( plgin );
            char buff[ 256 ];
            plgin->GetFilterPluginInfo( buff, sizeof( buff ));
            m_vPlginNames.push_back( buff );
        }else{
            delete plgin;
        }

    }

    delete PlginPath;
    delete wk;


    return 1;
}
int FreePlugins()
{
    vector< CImageFilterPlgin* >::iterator it = m_vPlgins.begin();
    for( ; it != m_vPlgins.end(); it++){
        delete *it;
    }
    m_vPlgins.clear();
    m_vPlginNames.clear();
```



```
    return 1;
}

inline CImageFilterPlgin *operator [] (int index ){ return m_vPlgins[ index ]; }
inline void GetPluginName( int index, char *buff, int bufsiz ){
    strncpy( buff, m_vPlginNames[index].c_str(), bufsiz );
}

inline int GetPluginCounts(){   return m_vPlgins.size();    }
};



typedef int (WINAPI *GETRESIZEPLUGININFO)(LPSTR buf,int buflen);

typedef BYTE* (WINAPI *DORESIZE)( BITMAPINFO *bi, BYTE *imagedata,BITMAPINFO
*to_bi,FARPROC lpPrgressCallback,long lData);

class CImageResizePlgin
{
    HINSTANCE   hDLL;
    char        _plugin_path_name[ MAX_PATH ];

    GETRESIZEPLUGININFO     pGetResizePluginInfo;
    DORESIZE            pDoResize;

public:
    CImageResizePlgin(){
    }
    virtual ~CImageResizePlgin(){
        FreeLibrary( hDLL );
    }



    // プラグインローダー
    int LoadPlugin(const char* PluginName)
    {
        pGetResizePluginInfo    =   NULL;
        pDoResize       =   NULL;

        hDLL = ::LoadLibrary( PluginName );    // DLLのロード
        if (hDLL != NULL)
        {
            pGetResizePluginInfo    =   (GETRESIZEPLUGININFO)::GetProcAddress(hDLL,"G
etResizePluginInfo" );
            pDoResize   =   (DORESIZE)::GetProcAddress( hDLL, "DoResize" );
            if( pGetResizePluginInfo && pDoResize ){
                strncpy( _plugin_path_name, PluginName, MAX_PATH );
                return TRUE;
            }else{
                return FALSE;
            }
```

```
        }else{
            return FALSE;
        }

    }




    // プラグイン関数
    //
    int GetResizePluginInfo(LPSTR buf,int buflen)
    {
        if( pGetResizePluginInfo )
            return (*pGetResizePluginInfo)(  buf, buflen );
        else
            return 0;
    }

    BYTE* DoResize(BITMAPINFO *bi, BYTE *imagedata,BITMAPINFO *to_bi,FARPROC
lpPrgressCallback,long lData)
    {
        if( pDoResize )
            return (*pDoResize)( bi, imagedata,to_bi,lpPrgressCallback,lData );
        else
            return 0;
    }


};
class CImageResizePlginLoader
{
    vector< CImageResizePlgin* >        m_vPlgins;
    vector< string >                    m_vPlginNames;

public:
    CImageResizePlginLoader(){

    }
    virtual ~CImageResizePlginLoader(){
        FreePlugins();
    }


    int SetResizePluginDirectory( const char *path )
    {
        HANDLE              fh;
        WIN32_FIND_DATA     ff;
        char    *PlginPath, *wk;
        CImageResizePlgin   *plgin;
        vector< string >    vplgins;
```



```
        PlginPath = new char[ lstrlen( path ) + MAX_PATH ];
        wk = new char[ lstrlen(path ) + MAX_PATH ];

        sprintf( PlginPath, "%s\\*.dll", path );

        fh = FindFirstFile( PlginPath, &ff );
        while( fh != INVALID_HANDLE_VALUE ){
            sprintf( wk, "%s\\%s", path, ff.cFileName );
            vplgins.push_back( wk );
            if (FindNextFile(fh, &ff) == false) break;
        }

        //  プラグインをソートしてロード
        sort( vplgins.begin(), vplgins.end());
        vector<string>::iterator it_plgin = vplgins.begin();
        for( ; it_plgin != vplgins.end(); it_plgin++){
            plgin = new CImageResizePlgin();
            if( plgin->LoadPlugin( it_plgin->c_str() ) ){
                m_vPlgins.push_back( plgin );
                char buff[ 256 ];
                plgin->GetResizePluginInfo( buff, sizeof( buff ));
                m_vPlginNames.push_back( buff );
            }else{
                delete plgin;
            }

        }

        delete PlginPath;
        delete wk;


        return 1;
    }
    int FreePlugins()
    {
        vector< CImageResizePlgin* >::iterator it = m_vPlgins.begin();
        for( ; it != m_vPlgins.end(); it++){
            delete *it;
        }
        m_vPlgins.clear();
        m_vPlginNames.clear();

        return 1;
    }

    inline CImageResizePlgin *operator [] (int index ){ return m_vPlgins[ index ]; }
    inline void GetPluginName( int index, char *buff, int bufsiz ){
        strncpy( buff, m_vPlginNames[index].c_str(), bufsiz );
    }

    inline int GetPluginCounts(){   return m_vPlgins.size();    }
};
```

```
#endif
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by VideoEffector.rc
//
#define IDR_MANIFEST                    1
#define IDD_ABOUTBOX                    100
#define IDP_OLE_INIT_FAILED             100
#define IDR_MAINFRAME                   128
#define IDR_VideoEffectorTYPE           129
#define ID_MEDIAPLAYER_PLAY             130
#define ID_MEDIAPLAYER_PAUSE            131
#define ID_MEDIAPLAYER_STOP             132
#define ID_133                          133
#define ID_134                          134
#define ID_Menu                         135
#define ID_MEDIAPLAYER_DOUBLEPLAYBACKRATE 136
#define ID_MEDIAPLAYER_NORMALPLAYBACKRATE 137
#define ID_CAMERA_OPEN                  32776
#define ID_HEAD                         32781
#define ID_QUARTER                      32782
#define ID_HALF                         32783
#define ID_THREEQUARTER                 32784

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        139
#define _APS_NEXT_COMMAND_VALUE         32785
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

```
//-----------------------------------------------------------------------------
//
//  CMediaPlayCamera クラスヘッダ
//
//      CMediaPlayCamera.h
//
//  VC++.NET 2003               2004/11/01  THas
//
//-----------------------------------------------------------------------------
#ifndef __CMediaPlayCamera_h__
#define __CMediaPlayCamera_h__
//-----------------------------------------------------------------------------
//  include files
//-----------------------------------------------------------------------------

#include "CMediaPlayWnd.h"


//-----------------------------------------------------------------------------
//  definitions
//-----------------------------------------------------------------------------


//-----------------------------------------------------------------------------
//  CMediaPlayCamera class
//-----------------------------------------------------------------------------

class CMediaPlayCamera : public CMediaPlayWnd
{
    private:


    protected:


    public:

        //  constructors
        CMediaPlayCamera(HWND hWnd = NULL);

        //  destructors
        virtual ~CMediaPlayCamera();


        //  properties


        //  methods
        virtual HRESULT OpenClip(LPCTSTR pszDevName);
        virtual HRESULT OpenClip(LPCTSTR pszDevName, const GUID& Category);
        virtual HRESULT OpenClip(LPCTSTR pszDevName, LPCTSTR pszPinName);

        virtual HRESULT SetRate(double dRate = 1.0);
```



```
        virtual HRESULT SetPosPercent(double dPercent = 0.0);


    protected:

        //  helper functions
        HRESULT AddVideoInputDevice(LPCTSTR pszDevName, IBaseFilter **ppFilter);

};


//-----------------------------------------------------------------------------
#endif
```

```
//-----------------------------------------------------------------------------
//
//  CMediaPlayCamera クラス
//
//      CMediaPlayCamera.cpp
//
//  VC++.NET 2003               2004/11/01  THas
//
//-----------------------------------------------------------------------------
#include "CMediaPlayCamera.h"
//-----------------------------------------------------------------------------
//  include files
//-----------------------------------------------------------------------------

#include <atlbase.h>

#include "UtilDShow.h"


//-----------------------------------------------------------------------------
//  definitions
//-----------------------------------------------------------------------------



//-----------------------------------------------------------------------------
//  prototyps
//-----------------------------------------------------------------------------



//-----------------------------------------------------------------------------
//  implements
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
//  constructor

CMediaPlayCamera::CMediaPlayCamera(HWND hWnd)
: CMediaPlayWnd(hWnd)
{

}

//-----------------------------------------------------------------------------
//  destructor

CMediaPlayCamera::~CMediaPlayCamera()
{

}
```



```
//-----------------------------------------------------------------------------
//  public property interfaces



//-----------------------------------------------------------------------------
//  public methods

//-------------------------------------------------------------------
//  カメラデバイスを開く

//  デバイス名で開く
//-------------------------------------------------------------------
//  ピンは キャプチャピンを使う
//
//  引数
//      pszDevName : デバイス名(FriendlyName)
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayCamera::OpenClip(LPCTSTR pszDevName)
{
    return OpenClip(pszDevName, PIN_CATEGORY_CAPTURE);
}

//  デバイス名とピンカテゴリで開く
//-------------------------------------------------------------------
//
//  引数
//      pszDevName : デバイス名(FriendlyName)
//      Category   : ピンカテゴリ
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayCamera::OpenClip(LPCTSTR pszDevName, const GUID& Category)
{
    HRESULT hr;

    //  fool trap
    if(NULL == pszDevName)  return E_POINTER;

    //  close current clip
    CloseClip();

    //  create GraphBuilder
    hr = CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
                          IID_IGraphBuilder,
                          reinterpret_cast<LPVOID*>(&m_pIGraphBuilder));
    if(NULL == m_pIGraphBuilder)  return hr;

    //  add SampleGrabber
```

```
    hr = AddSampleGrabberFilter(MEDIATYPE_Video, MEDIASUBTYPE_RGB24,
FORMAT_VideoInfo);
    if(NULL == m_pISampleGrabber)  return hr;

    //  add source filter
    IBaseFilter *pSrc;
    hr = AddVideoInputDevice(pszDevName, &pSrc);
    if(NULL == pSrc)  return hr;

    //  render
    IPin    *pPin;
    hr = DS_GetPinByCategory(pSrc, Category, PINDIR_OUTPUT, &pPin);
    pSrc->Release();
    if(NULL == pPin)  return hr;
    m_pIGraphBuilder->Render(pPin);
    pPin->Release();

    //  get media type
    hr = m_pISampleGrabber->GetConnectedMediaType(&m_MediaType);
    if(S_OK != hr)  return hr;

    //  query interfaces
    QueryInterfaces();

    //  assign screen
    AssignScreen();

    //  setup audio
    MuteAudio();

    return S_OK;
}

//  デバイス名とピン名で開く
//------------------------------------------------------------------
//
//  引数
//      pszDevName : デバイス名(FriendlyName)
//      pszPinName : ピン名
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayCamera::OpenClip(LPCTSTR pszDevName, LPCTSTR pszPinName)
{
    HRESULT hr;

    //  fool trap
    if(NULL == pszDevName)  return E_POINTER;

    //  close current clip
    CloseClip();
```



```
    //  create GraphBuilder
    hr = CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
                          IID_IGraphBuilder,
                          reinterpret_cast<LPVOID*>(&m_pIGraphBuilder));
    if(NULL == m_pIGraphBuilder)  return hr;

    //  add SampleGrabber
    hr = AddSampleGrabberFilter(MEDIATYPE_Video, MEDIASUBTYPE_RGB24,
FORMAT_VideoInfo);
    if(NULL == m_pISampleGrabber)  return hr;

    //  add source filter
    IBaseFilter *pSrc;
    hr = AddVideoInputDevice(pszDevName, &pSrc);
    if(NULL == pSrc)  return hr;

    //  render
    IPin    *pPin;
    hr = DS_GetPin(pSrc, pszPinName, PINDIR_OUTPUT, &pPin);
    pSrc->Release();
    if(NULL == pPin)  return hr;
    hr = m_pIGraphBuilder->Render(pPin);
    pPin->Release();

    //  get media type
    hr = m_pISampleGrabber->GetConnectedMediaType(&m_MediaType);
    if(S_OK != hr)  return hr;

    //  query interfaces
    QueryInterfaces();

    //  assign screen
    AssignScreen();

    //  setup audio
    MuteAudio();

    return S_OK;
}

//  再生速度を設定する
//------------------------------------------------------------------
//  カメラは再生速度を指定できない
//
//  引数
//      dRate : 再生速度
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayCamera::SetRate(double dRate)
{
    return E_NOTIMPL;
```

```
}

//  再生位置を設定する
//---------------------------------------------------------------
//  カメラは再生位置を指定できない
//
//  引数
//      dPercent : 再生位置
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayCamera::SetPosPercent(double dPercent)
{
    return E_NOTIMPL;
}

//-----------------------------------------------------------------------------
//  helper functions


//  キャプチャデバイスをグラフに追加する
//---------------------------------------------------------------
//
//  引数
//      pszDevName : 追加するデバイス名
//      ppFilter   : 追加したフィルタを返す
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayCamera::AddVideoInputDevice(LPCTSTR pszDevName, IBaseFilter
**ppFilter)
{
    HRESULT hr;

    //  fool trap
    if(NULL == ppFilter)  return E_POINTER;
    *ppFilter = NULL;
    if(NULL == pszDevName)  return E_POINTER;
    if(NULL == m_pIGraphBuilder)  return E_ACCESSDENIED;

    hr = DS_GetVideoInputFilter(pszDevName, ppFilter);
    if(NULL == *ppFilter)  return hr;

    USES_CONVERSION;
    hr = m_pIGraphBuilder->AddFilter(*ppFilter, T2W(pszDevName));

    return hr;
}
```

```
//-----------------------------------------------------------------------------
//
//  CMediaPlayFile クラスヘッダ
//
//      CMediaPlayFile.h
//
//  VC++.NET 2003               2004/11/01  THas
//
//-----------------------------------------------------------------------------
#ifndef __CMediaPlayFile_h__
#define __CMediaPlayFile_h__
//-----------------------------------------------------------------------------
//  include files
//-----------------------------------------------------------------------------

#include "CMediaPlayWnd.h"


//-----------------------------------------------------------------------------
//  definitions
//-----------------------------------------------------------------------------


//-----------------------------------------------------------------------------
//  CMediaPlayFile class
//-----------------------------------------------------------------------------

class CMediaPlayFile : public CMediaPlayWnd
{
    private:


    protected:


    public:

        //  constructors
        CMediaPlayFile(HWND hWnd = NULL);

        //  destructors
        virtual ~CMediaPlayFile();


        //  properties


        //  methods
        virtual HRESULT OpenClip(LPCTSTR pszFileName);

};
```



```
//-----------------------------------------------------------------------------
#endif
```

```
//------------------------------------------------------------------------------
//
//  CMediaPlayFile クラス
//
//      CMediaPlayFile.cpp
//
//  VC++.NET 2003               2004/11/01  THas
//
//------------------------------------------------------------------------------
#include "CMediaPlayFile.h"
//------------------------------------------------------------------------------
//  include files
//------------------------------------------------------------------------------

#include <atlbase.h>


//------------------------------------------------------------------------------
//  definitions
//------------------------------------------------------------------------------



//------------------------------------------------------------------------------
//  prototyps
//------------------------------------------------------------------------------



//------------------------------------------------------------------------------
//  implements
//------------------------------------------------------------------------------

//------------------------------------------------------------------------------
//  constructor

CMediaPlayFile::CMediaPlayFile(HWND hWnd)
: CMediaPlayWnd(hWnd)
{

}

//------------------------------------------------------------------------------
//  destructor

CMediaPlayFile::~CMediaPlayFile()
{

}


//------------------------------------------------------------------------------
//  public property interfaces
```



```
//------------------------------------------------------------------------------
//  public methods

//  動画ファイルを開く
//------------------------------------------------------------------
//
//  引数
//      pszFileName : 動画ファイル名
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayFile::OpenClip(LPCTSTR pszFileName)
{
    USES_CONVERSION;

    HRESULT hr;

    //  fool trap
    if(NULL == pszFileName)  return E_POINTER;

    //  close current clip
    CloseClip();

    //  create GraphBuilder
    hr = CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
                          IID_IGraphBuilder,
                          reinterpret_cast<LPVOID*>(&m_pIGraphBuilder));
    if(NULL == m_pIGraphBuilder)  return hr;

    //  add SampleGrabber
    hr = AddSampleGrabberFilter(MEDIATYPE_Video, MEDIASUBTYPE_RGB24,
FORMAT_VideoInfo);
    if(NULL == m_pISampleGrabber)  return hr;

    //  add source file and render
    hr = m_pIGraphBuilder->RenderFile(T2W(pszFileName), NULL);
//    if(S_OK != hr)  return hr;    //  音声OFF時に  hr == 262744 となる  2004/12/28
THas
    if(FAILED(hr))  return hr;

    //  get media type
    hr = m_pISampleGrabber->GetConnectedMediaType(&m_MediaType);
    if(S_OK != hr)  return hr;

    //  query interfaces
    QueryInterfaces();

    //  assign screen
    AssignScreen();
```

```
    //  setup audio
    MuteAudio();

    return S_OK;
}


//-----------------------------------------------------------------------------
```

```
//-----------------------------------------------------------------------------
//
//  CMediaPlayWnd クラスヘッダ
//
//      CMediaPlayWnd.h
//
//  VC++.NET 2003               2004/11/01  THas
//
//-----------------------------------------------------------------------------
#pragma once
#ifndef __CMediaPlayWnd_h__
#define __CMediaPlayWnd_h__
//-----------------------------------------------------------------------------
//  include files
//-----------------------------------------------------------------------------

#include <dshow.h>

#include "CSampleGrabberCB.h"


//-----------------------------------------------------------------------------
//  definitions
//-----------------------------------------------------------------------------



//-----------------------------------------------------------------------------
//  CMediaPlayWnd class
//-----------------------------------------------------------------------------

class CMediaPlayWnd
{
    private:


    protected:

        HWND    m_hWndParent;

        IGraphBuilder   *m_pIGraphBuilder;
        ISampleGrabber  *m_pISampleGrabber;
        IMediaControl   *m_pIMediaControl;
        IMediaSeeking   *m_pIMediaSeeking;
        IMediaPosition  *m_pIMediaPosition;
        IMediaEventEx   *m_pIMediaEvent;
        IVideoWindow    *m_pIVideoWindow;

        AM_MEDIA_TYPE   m_MediaType;
        CSampleGrabberCB    m_SampleGrabberCB;
```



```
    public:

        //  constructor
        CMediaPlayWnd(HWND hWnd = NULL);

        //  destructor
        virtual ~CMediaPlayWnd();

        //  properties
        long    GetBitmapWidth(void)  const;
        long    GetBitmapHeight(void) const;
        HRESULT GetBitmapInfo(ULONG *pSize, BITMAPINFO *pDst) const;
        long    GetSampleSize(void) const;
        HRESULT GetCurrentBuffer(long *pSize, long *pBuff) const;
        HRESULT SetBufferCB(BUFFERCBFUNCTION pFunc, void *pData = NULL);

        //  methods
        virtual HRESULT OpenClip(LPCTSTR pszFileName);
        virtual HRESULT CloseClip(void);
        virtual HRESULT Play(void);
        virtual HRESULT Pause(void);
        virtual HRESULT Stop(void);
        virtual HRESULT SetRate(double dRate = 1.0);
        virtual HRESULT SetPosPercent(double dPercent = 0.0);

        void MoveWindow(int x = 0, int y = 0);
        void MoveWindow(int x, int y, int w, int h);


    protected:

        //  helper functions
        HRESULT AddSampleGrabberFilter(GUID majortype, GUID subtype, GUID
formattype);
        HRESULT QueryInterfaces(void);
        HRESULT ReleaseInterfaces(void);
        HRESULT AssignScreen(void);
        HRESULT MuteAudio(void);


};


//-----------------------------------------------------------------------------
#endif
```

```
//-----------------------------------------------------------------------------
//
//  CMediaPlayWnd クラス
//
//      CMediaPlayWnd.cpp
//
//  VC++.NET 2003               2004/11/01  THas
//
//-----------------------------------------------------------------------------
#include "CMediaPlayWnd.h"
//-----------------------------------------------------------------------------
//  include files
//-----------------------------------------------------------------------------

#include <atlbase.h>
#include <math.h>

#include "UtilDShow.h"

#pragma comment(lib,"strmbase.lib")
#pragma comment(lib,"strmiids.lib")
#pragma comment(lib,"quartz.lib")

//-----------------------------------------------------------------------------
//  definitions
//-----------------------------------------------------------------------------



//-----------------------------------------------------------------------------
//  prototyps
//-----------------------------------------------------------------------------



//-----------------------------------------------------------------------------
//  implements
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
//  constructor

CMediaPlayWnd::CMediaPlayWnd(HWND hWnd)
{
    m_hWndParent = hWnd;

    m_pIGraphBuilder  = NULL;
    m_pISampleGrabber = NULL;
    m_pIMediaControl  = NULL;
    m_pIMediaSeeking  = NULL;
    m_pIMediaPosition = NULL;
    m_pIMediaEvent    = NULL;
    m_pIVideoWindow   = NULL;
```



```
    memset(&m_MediaType, 0x00, sizeof(AM_MEDIA_TYPE));
}

//-----------------------------------------------------------------------------
//  destructor

CMediaPlayWnd::~CMediaPlayWnd()
{
    CloseClip();
}


//-----------------------------------------------------------------------------
//  public property interfaces

//  画像の大きさを返す
//-------------------------------------------------------------------

//  画像の幅を返す
long CMediaPlayWnd::GetBitmapWidth() const
{
    if( (FORMAT_VideoInfo == m_MediaType.formattype) &&
        (NULL != m_MediaType.pbFormat) ) {

        VIDEOINFOHEADER     *pvdh;
        BITMAPINFOHEADER    *pbmh;

        pvdh = reinterpret_cast<VIDEOINFOHEADER*>(m_MediaType.pbFormat);
        pbmh = &(pvdh->bmiHeader);

        return pbmh->biWidth;
    } else {
        return 0;
    }
}

//  画像の高さを返す
long CMediaPlayWnd::GetBitmapHeight() const
{
    if( (FORMAT_VideoInfo == m_MediaType.formattype) &&
        (NULL != m_MediaType.pbFormat) ) {

        VIDEOINFOHEADER     *pvdh;
        BITMAPINFOHEADER    *pbmh;

        pvdh = reinterpret_cast<VIDEOINFOHEADER*>(m_MediaType.pbFormat);
        pbmh = &(pvdh->bmiHeader);

        return pbmh->biHeight;
    } else {
        return 0;
    }
```

```
}

// 画像の情報を返す
//---------------------------------------------------------------
//
// 引数
//      pSize : pDst のサイズ
//              pDst が NULL の時は BITMAPINFO を受け取るのに必要なサイズを返す
//      pDst : BITMAPINFO を返す
//
// 戻り値
//      エラーコードを返す
//          S_OK            : 成功
//          E_POINTER       : pSize が NULL
//          E_OUTOFMEMORY   : 指定されたバッファの大きさが不十分
//          E_ACCESSDENIED : メディアが準備されていないか BITMAPINFO を持っていない
//
HRESULT CMediaPlayWnd::GetBitmapInfo(ULONG *pSize, BITMAPINFO *pDst) const
{
    if(NULL == pSize)  return E_POINTER;

    if( (FORMAT_VideoInfo == m_MediaType.formattype) &&
        (NULL != m_MediaType.pbFormat) ) {

        VIDEOINFOHEADER     *pvdh;
        BITMAPINFO          *pbmi;
        ULONG               bmi_size;


        pvdh = reinterpret_cast<VIDEOINFOHEADER*>(m_MediaType.pbFormat);
        pbmi = reinterpret_cast<BITMAPINFO*>(&(pvdh->bmiHeader));
        bmi_size = m_MediaType.cbFormat - sizeof(VIDEOINFOHEADER) +
sizeof(BITMAPINFOHEADER);

        if(NULL != pDst) {
            if(*pSize < bmi_size)  return E_OUTOFMEMORY;
            memcpy(pDst, pbmi, bmi_size);
        }
        *pSize = bmi_size;

        return S_OK;
    } else {
        return E_ACCESSDENIED;
    }
}

// 画像 1 フレーム分のデータサイズを返す
//---------------------------------------------------------------
//
// 引数
//      なし
//
// 戻り値
```



```
//      データサイズを返す (long 単位)
//
long CMediaPlayWnd::GetSampleSize() const
{
    return m_MediaType.lSampleSize;
}

// 1 フレーム分のデータを返す
//---------------------------------------------------------------
// ISampleGrabber::GetCurrentBuffer() を呼ぶ
//      pBuff に NULL を指定すると pSize に必要なバッファサイズが得られる
//      … 筈だが 上手く機能しない様なので GetSampleSize() の使用を推奨
//
// 引数
//      pSize : pBuff のサイズ
//      pBuff : 画像データを返す
//
// 戻り値
//      エラーコードを返す
HRESULT CMediaPlayWnd::GetCurrentBuffer(long *pSize, long *pBuff) const
{
    if(NULL == m_pISampleGrabber) {
        return E_ACCESSDENIED;
    } else {
        return m_pISampleGrabber->GetCurrentBuffer(pSize, pBuff);
    }
}

// サンプル到着時に呼び出すコールバック関数を指定する
//---------------------------------------------------------------
// ISampleGrabber::SetCallback() を呼ぶ
//
// 引数
//      pFunc : 設定するコールバック関数
//      pData : ??? 親オブジェクトのポインタを渡している様子
//
// 戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::SetBufferCB(BUFFERCBFUNCTION pFunc, void *pData)
{
    if(NULL == m_pISampleGrabber) {
        return E_ACCESSDENIED;
    } else {
        HRESULT hr;

        m_pISampleGrabber->SetBufferSamples(FALSE);
        m_pISampleGrabber->SetCallback(NULL, 1);
        m_SampleGrabberCB.SetBufferCBFunction(pFunc, pData);
        hr = m_pISampleGrabber->SetCallback(&m_SampleGrabberCB, 1);
        m_pISampleGrabber->SetBufferSamples(TRUE);

        return hr;
```

```
    }
}


//-----------------------------------------------------------------------------
//  public methods

//  ストリームを開く
//-----------------------------------------------------------------
//  派生クラスのためのプレースホルダ
//
HRESULT CMediaPlayWnd::OpenClip(LPCTSTR pszFileName)
{
    return E_NOTIMPL;
}

//  ストリームを閉じる
//-----------------------------------------------------------------
//
//  引数
//      なし
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::CloseClip()
{
    //  停止
    if(NULL != m_pISampleGrabber) {
        m_pISampleGrabber->SetBufferSamples(FALSE);
        m_pISampleGrabber->SetCallback(NULL, 1);
    }

    if(NULL != m_pIMediaControl) {
        m_pIMediaControl->Stop();
    }

    if(NULL != m_pIVideoWindow) {
        m_pIVideoWindow->put_Visible(OAFALSE);
        m_pIVideoWindow->put_Owner(NULL);
    }

    //  リソース解放
    ReleaseInterfaces();
    DS_FreeMediaType(&m_MediaType);
    memset(&m_MediaType, 0x00, sizeof(AM_MEDIA_TYPE));

    return S_OK;
}

//  再生する
//-----------------------------------------------------------------
//
```



```
//  引数
//      なし
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::Play()
{
    if(NULL == m_pIMediaControl) {
        return E_ACCESSDENIED;
    } else {
        if(NULL != m_pISampleGrabber) {
            m_pISampleGrabber->SetBufferSamples(TRUE);
        }
        return m_pIMediaControl->Run();
    }
}

//  一時停止する
//-----------------------------------------------------------------
//
//  引数
//      なし
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::Pause()
{
    if(NULL == m_pIMediaControl) {
        return E_ACCESSDENIED;
    } else {
        return m_pIMediaControl->Pause();
    }
}

//  停止する
//-----------------------------------------------------------------
//
//  引数
//      なし
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::Stop()
{
    if(NULL == m_pIMediaControl) {
        return E_ACCESSDENIED;
    } else {
        if(NULL != m_pISampleGrabber) {
            m_pISampleGrabber->SetBufferSamples(FALSE);
        }
```

```
        m_pIMediaControl->Pause();
        if(NULL != m_pIMediaSeeking) {
            LONGLONG pos =0;
            m_pIMediaSeeking->SetPositions(&pos, AM_SEEKING_AbsolutePositioning,
                                           NULL, AM_SEEKING_NoPositioning);
        }
        return m_pIMediaControl->Stop();
    }
}

//  再生速度を設定する
//------------------------------------------------------------------
//
//  引数
//      dRate : 再生速度
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::SetRate(double dRate)
{
    if(NULL == m_pIMediaSeeking) {
        return E_ACCESSDENIED;
    } else {
        if(fabs(dRate) <= DBL_MIN)  return E_INVALIDARG;
        return m_pIMediaSeeking->SetRate(dRate);
    }
}

//  再生位置を設定する
//------------------------------------------------------------------
//
//  引数
//      dPercent : 再生位置
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::SetPosPercent(double dPercent)
{
    if(NULL == m_pIMediaPosition) {
        return E_ACCESSDENIED;
    } else {
        if(dPercent < 0.0)  return E_INVALIDARG;

        REFTIME pos;
        REFTIME duration;

        m_pIMediaPosition->get_Duration(&duration);
        pos = duration / 100.0 * dPercent;
        return m_pIMediaPosition->put_CurrentPosition(pos);
    }
}
```



```
//  表示位置を変更する
//------------------------------------------------------------------
//
//  引数
//      x : 基準点(左上)の横位置
//      y : 基準点(左上)の縦位置
//
//  戻り値
//      なし
//
void CMediaPlayWnd::MoveWindow(int x, int y)
{
    if(NULL != m_pIVideoWindow) {
        long    w, h;
        m_pIVideoWindow->get_Width(&w);
        m_pIVideoWindow->get_Height(&h);
        m_pIVideoWindow->SetWindowPosition(x, y, w, h);
    }
    return;
}

//  表示位置を変更する
//------------------------------------------------------------------
//
//  引数
//      x : 基準点(左上)の横位置
//      y : 基準点(左上)の縦位置
//      w : 表示領域の幅
//      h : 表示領域の高さ
//
//  戻り値
//      なし
//
void CMediaPlayWnd::MoveWindow(int x, int y, int w, int h)
{
    if(NULL != m_pIVideoWindow) {
        m_pIVideoWindow->SetWindowPosition(x, y, w, h);
    }
    return;
}

//------------------------------------------------------------------------------
//  helper functions

//  グラフにサンプルグラバフィルタを追加する
//------------------------------------------------------------------
//
//  引数
//      majortype  : メディアサンプルのメジャータイプ
//      subtype    : メディアサンプルのサブタイプ
//      formattype : メディアサンプルのフォーマットタイプ
//
```

```
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::AddSampleGrabberFilter(GUID majortype, GUID subtype, GUID
formattype)
{
    HRESULT hr;

    if(NULL == m_pIGraphBuilder)  return E_FAIL;

    hr = CoCreateInstance(CLSID_SampleGrabber, NULL, CLSCTX_INPROC_SERVER,
                          IID_ISampleGrabber,
                          reinterpret_cast<LPVOID*>(&m_pISampleGrabber));
    if(NULL != m_pISampleGrabber) {

        AM_MEDIA_TYPE   mt;

        memset(&mt, 0x00, sizeof(AM_MEDIA_TYPE));
        mt.majortype  = majortype;
        mt.subtype    = subtype;
        mt.formattype = formattype;
        hr = m_pISampleGrabber->SetMediaType(&mt);
        if(S_OK == hr) {
            IBaseFilter *pGrb;
            hr = m_pISampleGrabber->QueryInterface(IID_IBaseFilter,
                                                   reinterpret_cast<LPVOID*>(&pGrb));
            if(NULL != pGrb) {
                hr = m_pIGraphBuilder->AddFilter(pGrb, L"SampleGrabber");
                pGrb->Release();
            }
        }

    }

    return hr;
}

//  各インターフェイスを準備する
//-------------------------------------------------------------------
//
//  引数
//      なし
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::QueryInterfaces()
{
    HRESULT hr;
    HRESULT result = S_OK;

    hr = m_pIGraphBuilder->QueryInterface(IID_IMediaControl,
```



```
reinterpret_cast<PVOID*>(&m_pIMediaControl));
    if( (S_OK == result) && (S_OK != result) )  result = hr;
    hr = m_pIGraphBuilder->QueryInterface(IID_IMediaSeeking,

reinterpret_cast<PVOID*>(&m_pIMediaSeeking));
    if( (S_OK == result) && (S_OK != result) )  result = hr;
    hr = m_pIGraphBuilder->QueryInterface(IID_IMediaPosition,

reinterpret_cast<PVOID*>(&m_pIMediaPosition));
    if( (S_OK == result) && (S_OK != result) )  result = hr;
    hr = m_pIGraphBuilder->QueryInterface(IID_IMediaEvent,
                                          reinterpret_cast<PVOID*>(&m_pIMediaEvent));
    if( (S_OK == result) && (S_OK != result) )  result = hr;
    hr = m_pIGraphBuilder->QueryInterface(IID_IVideoWindow,

reinterpret_cast<PVOID*>(&m_pIVideoWindow));
    if( (S_OK == result) && (S_OK != result) )  result = hr;

    return result;
}

//  各インターフェイスを解放する
//-------------------------------------------------------------------
//
//  引数
//      なし
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::ReleaseInterfaces()
{

    if(NULL != m_pIVideoWindow) {
         m_pIVideoWindow->Release();
        m_pIVideoWindow = NULL;
    }

    if(NULL != m_pIMediaEvent) {
        m_pIMediaEvent->Release();
        m_pIMediaEvent = NULL;
    }

    if(NULL != m_pIMediaPosition) {
        m_pIMediaPosition->Release();
        m_pIMediaPosition = NULL;
    }

    if(NULL != m_pIMediaSeeking) {
        m_pIMediaSeeking->Release();
        m_pIMediaSeeking = NULL;
    }
```

```
    if(NULL != m_pIMediaControl) {
        m_pIMediaControl->Release();
        m_pIMediaControl = NULL;
    }

    if(NULL != m_pISampleGrabber) {
        m_pISampleGrabber->Release();
        m_pISampleGrabber = NULL;
    }

    if(NULL != m_pIGraphBuilder) {
        m_pIGraphBuilder->Release();
        m_pIGraphBuilder = NULL;
    }

    return S_OK;
}

//  プレビュー画面を設定する
//------------------------------------------------------------
//
//  引数
//      なし
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::AssignScreen()
{
    if(NULL == m_pIVideoWindow)  return E_ACCESSDENIED;

    if(NULL == m_hWndParent) {
        USES_CONVERSION;
        m_pIVideoWindow->put_Caption(T2BSTR(TEXT("Live")));
    } else {
        long    w, h;

        m_pIVideoWindow->put_Owner(reinterpret_cast<OAHWND>(m_hWndParent));
        m_pIVideoWindow->put_WindowStyle(WS_CHILD | WS_CLIPSIBLINGS);

        m_pIVideoWindow->get_Width(&w);
        m_pIVideoWindow->get_Height(&h);
        m_pIVideoWindow->SetWindowPosition(0, 0, w, h);
    }
    m_pIVideoWindow->put_Visible(OATRUE);

    return S_OK;
}

//  音声をミュートする
//------------------------------------------------------------
//
//  引数
```



```
//      なし
//
//  戻り値
//      エラーコードを返す
//
HRESULT CMediaPlayWnd::MuteAudio()
{
    if(NULL == m_pIGraphBuilder) {
        return E_ACCESSDENIED;
    } else {

        HRESULT     hr;
        IBasicAudio *pAudio;

        hr = m_pIGraphBuilder->QueryInterface(IID_IBasicAudio,
                                              reinterpret_cast<PVOID*>(&pAudio));
        if(NULL != pAudio) {
            hr = pAudio->put_Volume(-10000L);
            pAudio->Release();
        }

        return hr;
    }
}


//------------------------------------------------------------------------------
//
//------------------------------------------------------------------------------


//------------------------------------------------------------------------------
```

```
//-----------------------------------------------------------------------------
//
//  スナップショットダイアログ ヘッダ
//
//      DlgSnapshot.h
//
//  VC++.NET 2003                       2004/11/01  THas
//
//-----------------------------------------------------------------------------
#pragma once
//-----------------------------------------------------------------------------
//  includes
//-----------------------------------------------------------------------------

#include "CMediaPlayWnd.h"


//-----------------------------------------------------------------------------
//  definition
//-----------------------------------------------------------------------------

#ifndef WM_USER_SNAPSHOT
#define WM_USER_SNAPSHOT    WM_USER + 100
    //  wParam : フレームバッファのサイズ
    //  lParam : フレームバッファのポインタ
#endif


//-----------------------------------------------------------------------------
//  CDlgDeviceSelect class
//-----------------------------------------------------------------------------

// CDlgSnapshot ダイアログ

class CDlgSnapshot : public CDialog
{
    DECLARE_DYNCREATE(CDlgSnapshot)

public:
    CDlgSnapshot(CWnd* pParent = NULL);   // 標準コンストラクタ
    virtual ~CDlgSnapshot();

// ダイアログ データ
//  enum { IDD = IDD_DLG_SNAPSHOT };

protected:
//  virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート

    //  events
    virtual BOOL OnInitDialog();


    //  variables
```



```
    ULONG               m_nBitmapInfoSize;
    BITMAPINFO          *m_pBitmapInfo;
    long                m_nFrameBuffSize;
    BYTE                *m_pFrameBuff;


public:

    //  properties
//    void SetBtnSnapshotVisible(BOOL Value);


    //  methods
    BOOL SetBitmapInfo(CMediaPlayWnd *pMediaPlayer);
    BOOL AllocFrameBuff(long nSize);
    BOOL SetFrameData(long nSize, BYTE *pData);


private:

    //  helper function
    BOOL DrawFrame(BYTE *pData);


/*  DECLARE_MESSAGE_MAP()
public:
    afx_msg void OnBnClickedOk();
    afx_msg void OnBnClickedCancel();
    afx_msg void OnBnClickedBtnSnapshot();
    afx_msg void OnPaint();
    */
};
```

```
//-----------------------------------------------------------------------------
//
//  スナップショットダイアログ 実装ファイル
//
//      DlgSnapshot.cpp
//
//  VC++.NET 2003                       2004/11/01  THas
//
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
//  includes
//-----------------------------------------------------------------------------

#include "stdafx.h"
//#include "CaptureSample.h"
#include "DlgSnapshot.h"
//#include ".\dlgsnapshot.h"


//-----------------------------------------------------------------------------
//  prototypes
//-----------------------------------------------------------------------------



//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------

// CDlgSnapshot ダイアログ

//IMPLEMENT_DYNCREATE(CDlgSnapshot, CDHtmlDialog)
IMPLEMENT_DYNCREATE(CDlgSnapshot, CDialog)

CDlgSnapshot::CDlgSnapshot(CWnd* pParent /*=NULL*/)
//  : CDialog(CDlgSnapshot::IDD)
{
    //  モジュール変数初期化
    m_nBitmapInfoSize = 0;
    m_pBitmapInfo     = NULL;
    m_nFrameBuffSize  = 0;
    m_pFrameBuff      = NULL;
}

CDlgSnapshot::~CDlgSnapshot()
{
    //  モジュール変数クリア
    if(NULL != m_pBitmapInfo) {
        delete[] m_pBitmapInfo;
        //m_pBitmapInfo = NULL;
    }
    //m_nBitmapInfoSize = 0;
    if(NULL != m_pFrameBuff) {
```



```
        delete[] m_pFrameBuff;
        //m_pFrameBuff = NULL;
    }
    //m_nFrameBuffSize  = 0;
}

/*void CDlgSnapshot::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
}
*/

//-----------------------------------------------------------------------------
//  events (override)
//-----------------------------------------------------------------------------

//  ダイアログ初期化
BOOL CDlgSnapshot::OnInitDialog()
{
    CDialog::OnInitDialog();

    return TRUE;
}


//-----------------------------------------------------------------------------
//  public properties
//-----------------------------------------------------------------------------

//  スナップショットボタンの可視・不可視を設定する
//-----------------------------------------------------------------
//
//  引数
//      Value : ボタンの状態
//
//  戻り値
//      なし
//
/*void CDlgSnapshot::SetBtnSnapshotVisible(BOOL Value)
{
    CWnd *pbtn = GetDlgItem(IDC_BTN_SNAPSHOT);
    if(NULL != pbtn) {
        if(Value)  pbtn->ShowWindow(SW_SHOW);
        else       pbtn->ShowWindow(SW_HIDE);
    }
    return;
}
*/
//-----------------------------------------------------------------------------
//  public methods
//-----------------------------------------------------------------------------

//  BITMAPINFO を設定する
```

```
//-------------------------------------------------------------
//
//  引数
//      pMediaPlayer : 動画を管理するオブジェクトのポインタ
//
//  戻り値
//      成功すれば TRUE
//      失敗すれば FALSE を返す
//
BOOL CDlgSnapshot::SetBitmapInfo(CMediaPlayWnd *pMediaPlayer)
{
    //  fool trap
    if(NULL == pMediaPlayer)  return FALSE;

    //  BITMAPINFO のサイズを調べる
    ULONG   bmi_size;
    pMediaPlayer->GetBitmapInfo(&bmi_size, NULL);

    //  必要なら保存領域を確保する
    if(m_nBitmapInfoSize < bmi_size) {
        if(NULL != m_pBitmapInfo) {
            delete[] reinterpret_cast<BYTE*>(m_pBitmapInfo);
            m_nBitmapInfoSize = 0;
        }
        if(0 < bmi_size) {
            m_pBitmapInfo = reinterpret_cast<BITMAPINFO*>(new BYTE[bmi_size]);
            if(NULL == m_pBitmapInfo)  return FALSE;
            m_nBitmapInfoSize = bmi_size;
        }
    }

    //  BITMAPINFO を取得する
    if(S_OK == pMediaPlayer->GetBitmapInfo(&m_nBitmapInfoSize, m_pBitmapInfo)) {
        return TRUE;
    } else {
        return FALSE;
    }
}

//  フレームバッファを割り当てる
//-------------------------------------------------------------
//
//  引数
//      nSize : 割り当てるバッファサイズ(バイトサイズ)
//
//  戻り値
//      成功すれば TRUE
//      失敗すれば FALSE を返す
//
BOOL CDlgSnapshot::AllocFrameBuff(long nSize)
{
    //  fool trap
    if(nSize < 0)  return FALSE;
```



```
    //  既に確保されている場合は何もしない
    if(nSize == m_nFrameBuffSize) {
        //memset(m_pFrameBuff, 0x00, m_FrameBuffSize);
        return TRUE;
    }

    //  現在確保中の領域を解放する
    if(nSize != m_nFrameBuffSize) {
        if(NULL != m_pFrameBuff) {
            delete[] m_pFrameBuff;
            m_pFrameBuff = NULL;
        }
        m_nFrameBuffSize = 0;
    }

    //  新しい領域を確保する
    if(0 < nSize) {
        m_pFrameBuff = new BYTE[nSize];
        if(NULL == m_pFrameBuff) {
            return FALSE;
        }
        m_nFrameBuffSize = nSize;
        memset(m_pFrameBuff, 0x00, m_nFrameBuffSize);
    }

    return TRUE;
}

//  フレームデータを設定する
//-------------------------------------------------------------
//
//  引数
//      nSize : データのサイズ(バイトサイズ)
//      pData : 画像データのポインタ
//
//  戻り値
//      成功すれば TRUE
//      失敗すれば FALSE を返す
//
BOOL CDlgSnapshot::SetFrameData(long nSize, BYTE *pData)
{
    //  fool trap
    if(nSize < 0)      return FALSE;
    if(NULL == pData)  return FALSE;

    //  必要なら保存領域を確保する
    if(m_nFrameBuffSize < nSize) {
        AllocFrameBuff(nSize);
        if(nSize != m_nFrameBuffSize)  return FALSE;
    }

    //  データをコピーする
```

```
    if(0 < nSize) {
        memcpy(m_pFrameBuff, pData, nSize);
    }

    //  画面を描画する
    DrawFrame(m_pFrameBuff);

    return TRUE;
}


//-----------------------------------------------------------------------------
//  helper functions
//-----------------------------------------------------------------------------

//  フレーム(画像)を描画する
//-------------------------------------------------------------
//
//  引数
//      pData : 画像データのポインタ
//
//  戻り値
//      成功すれば TRUE
//      失敗すれば FALSE を返す
//
BOOL CDlgSnapshot::DrawFrame(BYTE *pData)
{
    //  fool trap
    if(NULL == pData)          return FALSE;
    if(NULL == m_pBitmapInfo)  return FALSE;

    //  draw frame
    BITMAPINFOHEADER *pbmh = reinterpret_cast<BITMAPINFOHEADER*>(m_pBitmapInfo);
    CDC              *pcdc = GetDC();
    SetDIBitsToDevice(pcdc->m_hDC, 0, 0,
                      pbmh->biWidth, pbmh->biHeight, 0, 0,
                      0, pbmh->biHeight,
                      pData,
                      m_pBitmapInfo,
                      DIB_RGB_COLORS);
    ReleaseDC(pcdc);

    return TRUE;
}

/*
//-----------------------------------------------------------------------------
//  message handlers
//-----------------------------------------------------------------------------

//  message map
//-------------------------------------------------------------
BEGIN_MESSAGE_MAP(CDlgSnapshot, CDialog)
```



```
    ON_BN_CLICKED(IDOK, OnBnClickedOk)
    ON_BN_CLICKED(IDCANCEL, OnBnClickedCancel)
    ON_WM_PAINT()
    ON_BN_CLICKED(IDC_BTN_SNAPSHOT, OnBnClickedBtnSnapshot)
END_MESSAGE_MAP()
//-----------------------------------------------------------------------------

// CDlgSnapshot メッセージ ハンドラ

//  OK ボタンクリック時
//-------------------------------------------------------------
void CDlgSnapshot::OnBnClickedOk()
{
    OnOK();
}

//  キャンセルボタンクリック時
//-------------------------------------------------------------
void CDlgSnapshot::OnBnClickedCancel()
{
    OnCancel();
}

//  スナップショットボタンクリック時
//-------------------------------------------------------------
void CDlgSnapshot::OnBnClickedBtnSnapshot()
{
    CWnd *pParent = GetParent();
    if(NULL != pParent) {
        pParent->PostMessage(WM_USER_SNAPSHOT,
                             static_cast<WPARAM>(m_nFrameBuffSize),
                             reinterpret_cast<LPARAM>(m_pFrameBuff));
    }
}

//  フォーム描画時
//-------------------------------------------------------------
void CDlgSnapshot::OnPaint()
{
    CPaintDC    dc(this);

    if( (NULL != m_pBitmapInfo) && (NULL != m_pFrameBuff) ) {
        BITMAPINFOHEADER *pbmh = reinterpret_cast<BITMAPINFOHEADER*>(m_pBitmapInfo);
        SetDIBitsToDevice(dc.m_hDC, 0, 0,
                          pbmh->biWidth, pbmh->biHeight, 0, 0,
                          0, pbmh->biHeight,
                          m_pFrameBuff,
                          m_pBitmapInfo,
                          DIB_RGB_COLORS);
    }
}
*/
```

```
//------------------------------------------------------------------------------
// File: DShowUtil.cpp
//
// Desc: DirectShow sample code - utility functions.
//
// Copyright (c) 2000-2001 Microsoft Corporation.  All rights reserved.
//------------------------------------------------------------------------------

#include "stdafx.h"
#include <atlbase.h>
#include <dshow.h>
#include <mtype.h>
#include <wxdebug.h>
#include <reftime.h>

#include "dshowutil.h"


HRESULT FindRenderer(IGraphBuilder *pGB, const GUID *mediatype, IBaseFilter
**ppFilter)
{
    HRESULT hr;
    IEnumFilters *pEnum = NULL;
    IBaseFilter *pFilter = NULL;
    IPin *pPin;
    ULONG ulFetched, ulInPins, ulOutPins;
    BOOL bFound=FALSE;

    // Verify graph builder interface
    if (!pGB)
        return E_NOINTERFACE;

    // Verify that a media type was passed
    if (!mediatype)
        return E_POINTER;

    // Clear the filter pointer in case there is no match
    if (ppFilter)
        *ppFilter = NULL;

    // Get filter enumerator
    hr = pGB->EnumFilters(&pEnum);
    if (FAILED(hr))
        return hr;

    pEnum->Reset();

    // Enumerate all filters in the graph
    while(!bFound && (pEnum->Next(1, &pFilter, &ulFetched) == S_OK))
    {
#ifdef DEBUG
        // Read filter name for debugging purposes
        FILTER_INFO FilterInfo;
```



```
        TCHAR szName[256];

        hr = pFilter->QueryFilterInfo(&FilterInfo);
        if (SUCCEEDED(hr))
        {
            // Show filter name in debugger
#ifdef UNICODE
            lstrcpy(szName, FilterInfo.achName);
#else
            WideCharToMultiByte(CP_ACP, 0, FilterInfo.achName, -1, szName, 256, 0,
0);
#endif
            FilterInfo.pGraph->Release();
        }
#endif

        // Find a filter with one input and no output pins
        hr = CountFilterPins(pFilter, &ulInPins, &ulOutPins);
        if (FAILED(hr))
            break;

        if ((ulInPins == 1) && (ulOutPins == 0))
        {
            // Get the first pin on the filter
            pPin=0;
            pPin = GetInPin(pFilter, 0);

            // Read this pin's major media type
            AM_MEDIA_TYPE type={0};
            hr = pPin->ConnectionMediaType(&type);
            if (FAILED(hr))
                break;

            // Is this pin's media type the requested type?
            // If so, then this is the renderer for which we are searching.
            // Copy the interface pointer and return.
            if (type.majortype == *mediatype)
            {
                // Found our filter
                *ppFilter = pFilter;
                bFound = TRUE;;
            }
            // This is not the renderer, so release the interface.
            else
                pFilter->Release();

            // Delete memory allocated by ConnectionMediaType()
            FreeMediaType(type);
        }
        else
        {
            // No match, so release the interface
            pFilter->Release();
```

```
        }
    }

    pEnum->Release();
    return hr;
}

HRESULT FindAudioRenderer(IGraphBuilder *pGB, IBaseFilter **ppFilter)
{
    return FindRenderer(pGB, &MEDIATYPE_Audio, ppFilter);
}

HRESULT FindVideoRenderer(IGraphBuilder *pGB, IBaseFilter **ppFilter)
{
    return FindRenderer(pGB, &MEDIATYPE_Video, ppFilter);
}

HRESULT CountFilterPins(IBaseFilter *pFilter, ULONG *pulInPins, ULONG *pulOutPins)
{
    HRESULT hr=S_OK;
    IEnumPins *pEnum=0;
    ULONG ulFound;
    IPin *pPin;

    // Verify input
    if (!pFilter || !pulInPins || !pulOutPins)
        return E_POINTER;

    // Clear number of pins found
    *pulInPins = 0;
    *pulOutPins = 0;

    // Get pin enumerator
    hr = pFilter->EnumPins(&pEnum);
    if(FAILED(hr))
        return hr;

    pEnum->Reset();

    // Count every pin on the filter
    while(S_OK == pEnum->Next(1, &pPin, &ulFound))
    {
        PIN_DIRECTION pindir = (PIN_DIRECTION)3;

        hr = pPin->QueryDirection(&pindir);

        if(pindir == PINDIR_INPUT)
            (*pulInPins)++;
        else
            (*pulOutPins)++;

        pPin->Release();
    }
```



```
    pEnum->Release();
    return hr;
}


HRESULT CountTotalFilterPins(IBaseFilter *pFilter, ULONG *pulPins)
{
    HRESULT hr;
    IEnumPins *pEnum=0;
    ULONG ulFound;
    IPin *pPin;

    // Verify input
    if (!pFilter || !pulPins)
        return E_POINTER;

    // Clear number of pins found
    *pulPins = 0;

    // Get pin enumerator
    hr = pFilter->EnumPins(&pEnum);
    if(FAILED(hr))
        return hr;

    // Count every pin on the filter, ignoring direction
    while(S_OK == pEnum->Next(1, &pPin, &ulFound))
    {
        (*pulPins)++;
        pPin->Release();
    }

    pEnum->Release();
    return hr;
}


HRESULT GetPin( IBaseFilter * pFilter, PIN_DIRECTION dirrequired, int iNum, IPin
**ppPin)
{
    CComPtr< IEnumPins > pEnum;
    *ppPin = NULL;
    HRESULT hr = pFilter->EnumPins(&pEnum);
    if(FAILED(hr))
        return hr;

    ULONG ulFound;
    IPin *pPin;
    hr = E_FAIL;

    while(S_OK == pEnum->Next(1, &pPin, &ulFound))
    {
        PIN_DIRECTION pindir = (PIN_DIRECTION)3;
```

```
        pPin->QueryDirection(&pindir);
        if(pindir == dirrequired)
        {
            if(iNum == 0)
            {
                *ppPin = pPin;
                // Found requested pin, so clear error
                hr = S_OK;
                break;
            }
            iNum--;
        }

        pPin->Release();
    }

    return hr;
}


IPin * GetInPin( IBaseFilter * pFilter, int Num )
{
    CComPtr< IPin > pComPin;
    GetPin(pFilter, PINDIR_INPUT, Num, &pComPin);
    return pComPin;
}


IPin * GetOutPin( IBaseFilter * pFilter, int Num )
{
    CComPtr< IPin > pComPin;
    GetPin(pFilter, PINDIR_OUTPUT, Num, &pComPin);
    return pComPin;
}


HRESULT FindOtherSplitterPin(IPin *pPinIn, GUID guid, int nStream, IPin **ppSplitPin)
{
    if (!ppSplitPin)
        return E_POINTER;

    CComPtr< IPin > pPinOut;
    pPinOut = pPinIn;

    while(pPinOut)
    {
        PIN_INFO ThisPinInfo;
        pPinOut->QueryPinInfo(&ThisPinInfo);
        if(ThisPinInfo.pFilter) ThisPinInfo.pFilter->Release();

        pPinOut = NULL;
        CComPtr< IEnumPins > pEnumPins;
        ThisPinInfo.pFilter->EnumPins(&pEnumPins);
```



```
        if(!pEnumPins)
        {
            return NULL;
        }

        // look at every pin on the current filter...
        //
        ULONG Fetched = 0;
        while(1)
        {
            CComPtr< IPin > pPin;
            Fetched = 0;
            ASSERT(!pPin); // is it out of scope?
            pEnumPins->Next(1, &pPin, &Fetched);
            if(!Fetched)
            {
                break;
            }

            PIN_INFO pi;
            pPin->QueryPinInfo(&pi);
            if(pi.pFilter) pi.pFilter->Release();

            // if it's an input pin...
            //
            if(pi.dir == PINDIR_INPUT)
            {
                // continue searching upstream from this pin
                //
                pPin->ConnectedTo(&pPinOut);

                // a pin that supports the required media type is the
                // splitter pin we are looking for!  We are done
                //
            }
            else
            {
                CComPtr< IEnumMediaTypes > pMediaEnum;
                pPin->EnumMediaTypes(&pMediaEnum);
                if(pMediaEnum)
                {
                    Fetched = 0;
                    AM_MEDIA_TYPE *pMediaType;
                    pMediaEnum->Next(1, &pMediaType, &Fetched);
                    if(Fetched)
                    {
                        if(pMediaType->majortype == guid)
                        {
                            if(nStream-- == 0)
                            {
                                DeleteMediaType(pMediaType);
                                *ppSplitPin = pPin;
                                (*ppSplitPin)->AddRef();
```

```
                                return S_OK;
                            }
                        }
                        DeleteMediaType(pMediaType);
                    }
                }
            }

            // go try the next pin

        } // while
    }
    ASSERT(FALSE);
    return E_FAIL;
}


HRESULT SeekNextFrame( IMediaSeeking * pSeeking, double FPS, long Frame )
{
    // try seeking by frames first
    //
    HRESULT hr = pSeeking->SetTimeFormat(&TIME_FORMAT_FRAME);
    REFERENCE_TIME Pos = 0;
    if(!FAILED(hr))
    {
        pSeeking->GetCurrentPosition(&Pos);
        Pos++;
    }
    else
    {
        // couldn't seek by frames, use Frame and FPS to calculate time
        //
        Pos = REFERENCE_TIME(double( Frame * UNITS ) / FPS);

        // add a half-frame to seek to middle of the frame
        //
        Pos += REFERENCE_TIME(double( UNITS ) * 0.5 / FPS);
    }

    hr = pSeeking->SetPositions(&Pos, AM_SEEKING_AbsolutePositioning,
                                NULL, AM_SEEKING_NoPositioning);
    return hr;

}

#ifdef DEBUG
    // for debugging purposes
    const INT iMAXLEVELS = 5;               // Maximum debug categories
    extern DWORD m_Levels[iMAXLEVELS];      // Debug level per category
#endif


void TurnOnDebugDllDebugging( )
```



```
{
#ifdef DEBUG
    for(int i = 0 ; i < iMAXLEVELS ; i++)
    {
        m_Levels[i] = 1;
    }
#endif
}

void DbgPrint( char * pText )
{
    DbgLog(( LOG_TRACE, 1, "%s", pText ));
}

void ErrPrint( char * pText )
{
    printf(pText);
    return;
}


// Adds a DirectShow filter graph to the Running Object Table,
// allowing GraphEdit to "spy" on a remote filter graph.
HRESULT AddGraphToRot(IUnknown *pUnkGraph, DWORD *pdwRegister)
{
    IMoniker * pMoniker;
    IRunningObjectTable *pROT;
    WCHAR wsz[128];
    HRESULT hr;

    if (FAILED(GetRunningObjectTable(0, &pROT)))
        return E_FAIL;

    wsprintfW(wsz, L"FilterGraph %08x pid %08x", (DWORD_PTR)pUnkGraph,
              GetCurrentProcessId());

    hr = CreateItemMoniker(L"!", wsz, &pMoniker);
    if (SUCCEEDED(hr))
    {
        hr = pROT->Register(0, pUnkGraph, pMoniker, pdwRegister);
        pMoniker->Release();
    }
    pROT->Release();
    return hr;
}

// Removes a filter graph from the Running Object Table
void RemoveGraphFromRot(DWORD pdwRegister)
{
    IRunningObjectTable *pROT;

    if (SUCCEEDED(GetRunningObjectTable(0, &pROT)))
    {
```

```
        pROT->Revoke(pdwRegister);
        pROT->Release();
    }
}


void ShowFilenameByCLSID(REFCLSID clsid, TCHAR *szFilename)
{
    HRESULT hr;
    LPOLESTR strCLSID;

    // Convert binary CLSID to a readable version
    hr = StringFromCLSID(clsid, &strCLSID);
    if(SUCCEEDED(hr))
    {
        TCHAR szKey[512];
        CString strQuery(strCLSID);

        // Create key name for reading filename registry
        wsprintf(szKey, TEXT("Software\\Classes\\CLSID\\%s\\InprocServer32\0"),
                 strQuery);

        // Free memory associated with strCLSID (allocated in StringFromCLSID)
        CoTaskMemFree(strCLSID);

        HKEY hkeyFilter=0;
        DWORD dwSize=MAX_PATH;
        BYTE szFile[MAX_PATH];
        int rc=0;

        // Open the CLSID key that contains information about the filter
        rc = RegOpenKey(HKEY_LOCAL_MACHINE, szKey, &hkeyFilter);
        if (rc == ERROR_SUCCESS)
        {
            rc = RegQueryValueEx(hkeyFilter, NULL,  // Read (Default) value
                                 NULL, NULL, szFile, &dwSize);

            if (rc == ERROR_SUCCESS)
                wsprintf(szFilename, TEXT("%s"), szFile);
            else
                wsprintf(szFilename, TEXT("<Unknown>\0"));

            rc = RegCloseKey(hkeyFilter);
        }
    }
}


HRESULT GetFileDurationString(IMediaSeeking *pMS, TCHAR *szDuration)
{
    HRESULT hr;

    if (!pMS)
```



```
        return E_NOINTERFACE;
    if (!szDuration)
        return E_POINTER;

    // Initialize the display in case we can't read the duration
    wsprintf(szDuration, TEXT("<00:00.000>"));

    // Is media time supported for this file?
    if (S_OK != pMS->IsFormatSupported(&TIME_FORMAT_MEDIA_TIME))
        return E_NOINTERFACE;

    // Read the time format to restore later
    GUID guidOriginalFormat;
    hr = pMS->GetTimeFormat(&guidOriginalFormat);
    if (FAILED(hr))
        return hr;

    // Ensure media time format for easy display
    hr = pMS->SetTimeFormat(&TIME_FORMAT_MEDIA_TIME);
    if (FAILED(hr))
        return hr;

    // Read the file's duration
    LONGLONG llDuration;
    hr = pMS->GetDuration(&llDuration);
    if (FAILED(hr))
        return hr;

    // Return to the original format
    if (guidOriginalFormat != TIME_FORMAT_MEDIA_TIME)
    {
        hr = pMS->SetTimeFormat(&guidOriginalFormat);
        if (FAILED(hr))
            return hr;
    }

    // Convert the LONGLONG duration into human-readable format
    unsigned long nTotalMS = (unsigned long) llDuration / 10000; // 100ns -> ms
    int nMS = nTotalMS % 1000;
    int nSeconds = nTotalMS / 1000;
    int nMinutes = nSeconds / 60;
    nSeconds %= 60;

    // Update the string
    wsprintf(szDuration, _T("%02dm:%02d.%03ds\0"), nMinutes, nSeconds, nMS);

    return hr;
}


BOOL SupportsPropertyPage(IBaseFilter *pFilter)
{
    HRESULT hr;
```

```
    ISpecifyPropertyPages *pSpecify;

    // Discover if this filter contains a property page
    hr = pFilter->QueryInterface(IID_ISpecifyPropertyPages, (void **)&pSpecify);
    if (SUCCEEDED(hr))
    {
        pSpecify->Release();
        return TRUE;
    }
    else
        return FALSE;
}


HRESULT ShowFilterPropertyPage(IBaseFilter *pFilter, HWND hwndParent)
{
    HRESULT hr;
    ISpecifyPropertyPages *pSpecify=0;

    if (!pFilter)
        return E_NOINTERFACE;

    // Discover if this filter contains a property page
    hr = pFilter->QueryInterface(IID_ISpecifyPropertyPages, (void **)&pSpecify);
    if (SUCCEEDED(hr))
    {
        do
        {
            FILTER_INFO FilterInfo;
            hr = pFilter->QueryFilterInfo(&FilterInfo);
            if (FAILED(hr))
                break;

            CAUUID caGUID;
            hr = pSpecify->GetPages(&caGUID);
            if (FAILED(hr))
                break;

            pSpecify->Release();

            // Display the filter's property page
            OleCreatePropertyFrame(
                hwndParent,             // Parent window
                0,                      // x (Reserved)
                0,                      // y (Reserved)
                FilterInfo.achName,     // Caption for the dialog box
                1,                      // Number of filters
                (IUnknown **)&pFilter,  // Pointer to the filter
                caGUID.cElems,          // Number of property pages
                caGUID.pElems,          // Pointer to property page CLSIDs
                0,                      // Locale identifier
                0,                      // Reserved
                NULL                    // Reserved
```



```
            );
            CoTaskMemFree(caGUID.pElems);
            FilterInfo.pGraph->Release();

        } while(0);
    }

    pFilter->Release();
    return hr;
}


//
// Some hardware decoders and video renderers support stepping media
// frame by frame with the IVideoFrameStep interface.  See the interface
// documentation for more details on frame stepping.
//
BOOL CanFrameStep(IGraphBuilder *pGB)
{
    HRESULT hr;
    IVideoFrameStep* pFS;

    hr = pGB->QueryInterface(__uuidof(IVideoFrameStep), (PVOID *)&pFS);
    if (FAILED(hr))
        return FALSE;

    // Check if this decoder can step
    hr = pFS->CanStep(0L, NULL);

    pFS->Release();

    if (hr == S_OK)
        return TRUE;
    else
        return FALSE;
}
```

```
#pragma once
#include <CDIB.h>


// CImageWnd

class CImageWnd : public CWnd
{
    DECLARE_DYNAMIC(CImageWnd)

public:
    CImageWnd();
    virtual ~CImageWnd();

    CDIB    DIB;
protected:
    DECLARE_MESSAGE_MAP()
public:
    afx_msg void OnPaint();
};
```

```
// ImageWnd.cpp : 実装ファイル
//

#include "stdafx.h"
#include "ImageWnd.h"


// CImageWnd

IMPLEMENT_DYNAMIC(CImageWnd, CWnd)
CImageWnd::CImageWnd()
{
    DIB.Create( 400,400);
    DIB.DrawCircle( 200,200,100,RGB(255,0,0));
}

CImageWnd::~CImageWnd()
{
}


BEGIN_MESSAGE_MAP(CImageWnd, CWnd)
    ON_WM_PAINT()
END_MESSAGE_MAP()



// CImageWnd メッセージ ハンドラ


void CImageWnd::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    if( DIB.Width() ){
        DIB.DrawDIBtoHDC( dc.m_hDC,0,0);

    }
}
```

```
// MainFrm.h : CMainFrame クラスのインターフェイス
//
#include "ImageFilterPlgin.h"

#pragma once
class CMainFrame : public CFrameWnd
{

protected: // シリアル化からのみ作成します。
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

// 属性
public:
    CImageFilterPlginLoader     plgins;
// 操作
public:

// オーバーライド
public:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

// 実装
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // コントロール バー用メンバ
    CStatusBar  m_wndStatusBar;
    CToolBar    m_wndToolBar;

// 生成された、メッセージ割り当て関数
protected:
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    DECLARE_MESSAGE_MAP()
public:
    int AddPlugins(void);
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO*
pHandlerInfo);

};
```

```
// MainFrm.cpp : CMainFrame クラスの実装
//

#include "stdafx.h"
#include "VideoEffector.h"
#include "VideoEffectorDoc.h"
#include "VideoEffectorView.h"


#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

#define     ID_FILTER_MENU_INDEX    (WM_USER + 10)

CMainFrame  *FrameWnd = NULL;

// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    ON_WM_CREATE()
END_MESSAGE_MAP()

static UINT indicators[] =
{
    ID_SEPARATOR,           // ステータス ライン インジケータ
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};


// CMainFrame コンストラクション/デストラクション

CMainFrame::CMainFrame()
{
    // TODO: メンバ初期化コードをここに追加してください。
    FrameWnd = this;
}

CMainFrame::~CMainFrame()
{
}


int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
```



```
    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP
        | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
        !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("ツールバーの作成に失敗しました。\n");
        return -1;      // 作成できませんでした。
    }

    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
          sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("ステータス バーの作成に失敗しました。\n");
        return -1;      // 作成できませんでした。
    }
    // TODO: ツールバーをドッキング可能にしない場合は、これらの 3 行を削除してください。
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);

    AddPlugins();

    return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    // TODO: この位置で CREATESTRUCT cs を修正して Window クラスまたはスタイルを
    //  修正してください。
    cs.cx = 730;
    cs.cy = 380;
    return TRUE;
}


// CMainFrame 診断

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG
```

```
// CMainFrame メッセージ ハンドラ


int CMainFrame::AddPlugins(void)
{
    char buff[ MAX_PATH ];
    ::GetModuleFileName( NULL, buff, sizeof( buff ));

    int len= lstrlen( buff );
    while( buff[ len ] != '\\' && len > 0 ){
        len--;
    }
    buff[len] = 0;
    strcat( buff, "\\plugins");
    plgins.SetFilterPluginDirectory( buff );



    CMenu   *pPlginMenu =GetMenu()->GetSubMenu( 5 );

    if( pPlginMenu == NULL )
        return 1;

    pPlginMenu->DeleteMenu(0,MF_BYPOSITION);


    int c = plgins.GetPluginCounts();

    for( int i = 0; i < c; i++){

        plgins.GetPluginName( i , buff, sizeof( buff ));
        pPlginMenu->AppendMenu(MF_STRING, ID_FILTER_MENU_INDEX + i, buff );
    }
    return 0;
}


BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO*
pHandlerInfo)
{
    if( pHandlerInfo == NULL )  //  すでに割り当てられているメッセージがない場合…
    {
        CVideoEffectorView  *pView = (CVideoEffectorView*)GetActiveView();


        if( nID >= ID_FILTER_MENU_INDEX  && nID < ID_FILTER_MENU_INDEX + 1000 ){
            if( nCode == CN_COMMAND ){
                pView->FilterNo = nID - ID_FILTER_MENU_INDEX;

            }else if( nCode == CN_UPDATE_COMMAND_UI ){
                //  シンボルをオンにします（これをしないとメニュー不可のままです）。
                CCmdUI* pCmdUI = (CCmdUI*)pExtra;
```



```
                pCmdUI->Enable( TRUE);
            }
            return TRUE;
        }
    }

    return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
}
```

```
#pragma once

#include <dshow.h>
#include <commctrl.h>
#include <commdlg.h>
#include <stdio.h>
#include <tchar.h>
#include <atlbase.h>
#include <support_functions.h>
#include <qedit.h>

//---C:\DXSDK\samples\Multimedia\DirectShow\BaseClasses---//
#include <Wxdebug.h>
#include <Combase.h>
#include <streams.h>
#include "dshowutil.h"

#include "CSampleGrabberCB.h"
//
// Macros
//
#define SAFE_RELEASE(x) { if (x) x->Release(); x = NULL; }


#define WM_GRAPHNOTIFY  WM_USER+13

// CMediaPlayWnd

class CMediaPlayWnd : public CWnd
{
    DECLARE_DYNAMIC(CMediaPlayWnd)

    DWORD     g_dwGraphRegister;//=0;

    // DirectShow interfaces
    IGraphBuilder *pGB;// = NULL;
    IMediaControl *pMC;// = NULL;
    IMediaEventEx *pME;// = NULL;
    IVideoWindow  *pVW;// = NULL;
    IBasicAudio   *pBA;// = NULL;
    IBasicVideo   *pBV;// = NULL;
    IMediaSeeking *pMS;// = NULL;
    IMediaPosition *pMP;// = NULL;
    IVideoFrameStep *pFS;// = NULL;
//  CComPtr< ISampleGrabber > pSG;// = NULL;
    ISampleGrabber      *pSG;


public:
    CMediaPlayWnd();
    virtual ~CMediaPlayWnd();

    long nWidth,nHeight;
```



```
    CSampleGrabberCB        CallBack;
protected:
    DECLARE_MESSAGE_MAP()
public:
    void Initialize(void);
    void CloseClip(void);
    HRESULT OpenClip(const char* FileName);
    HRESULT InitVideoWindow(int nMultiplier, int nDivider);
    void Play(void);
    void Pause(void);
    void Stop(void);
    void MoveVideoWindow(void);
    afx_msg void OnSize(UINT nType, int cx, int cy);
    BOOL GetFrameStepInterface(void);
    afx_msg void OnMove(int x, int y);
//  afx_msg void OnWindowPosChanged(WINDOWPOS* lpwndpos);
    HRESULT ModifyRate(double  dRateAdjust);
    HRESULT SetRate(double dRate);
    HRESULT GetDuration(REFTIME* pDuration);
    HRESULT SetPosPercent(int percent);
    afx_msg void OnPaint();
};
```

```
// MediaPlayWnd.cpp : 実装ファイル
//

#include "stdafx.h"
#include "MediaPlayWnd.h"

#pragma comment (lib, "strmbase.lib" )
#pragma comment( lib, "strmiids.lib")



#define REGISTER_FILTERGRAPH
// CMediaPlayWnd

IMPLEMENT_DYNAMIC(CMediaPlayWnd, CWnd)
CMediaPlayWnd::CMediaPlayWnd()
{
    CoInitialize(NULL);
    Initialize();
}

CMediaPlayWnd::~CMediaPlayWnd()
{
    CloseClip();

}


BEGIN_MESSAGE_MAP(CMediaPlayWnd, CWnd)
    ON_WM_SIZE()
    ON_WM_MOVE()
//  ON_WM_WINDOWPOSCHANGED()
ON_WM_PAINT()
END_MESSAGE_MAP()



// CMediaPlayWnd メッセージ ハンドラ
void CMediaPlayWnd::Initialize(void)
{
    // DirectShow interfaces
    pGB = NULL;
    pMC = NULL;
    pME = NULL;
    pVW = NULL;
    pBA = NULL;
    pBV = NULL;
    pMS = NULL;
    pMP = NULL;
    pFS = NULL;
    pSG = NULL;
}
```



```
void CMediaPlayWnd::CloseClip(void)
{
    HRESULT hr;

    // Stop SampleGrabber
    if( pSG ){
        pSG->SetCallback( NULL, 1 );
    }

    // Stop media playback
    if(pMC)
        hr = pMC->Stop();




    // Relinquish ownership (IMPORTANT!) after hiding video window
    if(pVW)
    {
        hr = pVW->put_Visible(OAFALSE);
        hr = pVW->put_Owner(NULL);
    }

    // Disable event callbacks
    if (pME)
        hr = pME->SetNotifyWindow((OAHWND)NULL, 0, 0);

#ifdef REGISTER_FILTERGRAPH
    if (g_dwGraphRegister)
    {
        RemoveGraphFromRot(g_dwGraphRegister);
        g_dwGraphRegister = 0;
    }
#endif

    // Release and zero DirectShow interfaces
    SAFE_RELEASE(pSG);
    SAFE_RELEASE(pGB);
    SAFE_RELEASE(pME);
    SAFE_RELEASE(pMS);
    SAFE_RELEASE(pMP);
    SAFE_RELEASE(pMC);
    SAFE_RELEASE(pBA);
    SAFE_RELEASE(pBV);
    SAFE_RELEASE(pVW);
    SAFE_RELEASE(pFS);
    SAFE_RELEASE(pGB);

    RECT rect;
    GetClientRect(&rect);
    InvalidateRect(&rect, TRUE);

}
```

```
HRESULT CMediaPlayWnd::OpenClip(const char* FileName)
{
    USES_CONVERSION;
    WCHAR wFile[MAX_PATH];
    HRESULT hr;

    CloseClip();

    // Clear open dialog remnants before calling RenderFile()
    UpdateWindow();

    // Convert filename to wide character string
    wcscpy(wFile, T2W(FileName));

    // Get the interface for DirectShow's GraphBuilder
    hr = CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
                          IID_IGraphBuilder, (void **)&pGB);
    if( FAILED( hr) ){
        MSGBOX(NULL,"Failed CoCreateInstance()" );
        return hr;
    }

    if(SUCCEEDED(hr))
    {
        // create a sample grabber
        //
        IBaseFilter     *pF = NULL;
        pSG = NULL;
        hr = CoCreateInstance(CLSID_SampleGrabber, NULL, CLSCTX_INPROC_SERVER,
                          IID_IBaseFilter,  (LPVOID*)&pF);
        pF->QueryInterface( IID_ISampleGrabber, (void**)&pSG);
        //hr = pSG.CoCreateInstance( CLSID_SampleGrabber );
        if( !pSG )
        {
            MSGBOX(NULL, TEXT("Could not create SampleGrabber (is qedit.dll
registered?)"));
            return hr;
        }

        CComQIPtr< IBaseFilter, &IID_IBaseFilter > pGrabBase( pSG );

        // force it to connect to video, 24 bit
        //
        AM_MEDIA_TYPE VideoType;
        ZeroMemory(&VideoType, sizeof(AM_MEDIA_TYPE));
        VideoType.majortype = MEDIATYPE_Video;
        VideoType.subtype = MEDIASUBTYPE_RGB24;
        VideoType.formattype = FORMAT_VideoInfo;
        hr = pSG->SetMediaType( &VideoType ); // shouldn't fail
        if( FAILED( hr ) )
        {
            MSGBOX(NULL, TEXT("Could not set media type"));
```



```
            return hr;
        }

        // add the grabber to the graph
        //
        hr = pGB->AddFilter( pGrabBase, L"Grabber" );
        if( FAILED( hr ) )
        {
            MSGBOX(NULL, TEXT("Could not put sample grabber in graph"));
            return hr;
        }


        // Have the graph builder construct its the appropriate graph automatically
        hr = pGB->RenderFile(wFile, NULL);
        if( FAILED( hr) ){
            MSGBOX(NULL,"Failed RenderFile()" );
            return hr;
        }



        AM_MEDIA_TYPE mt;
        hr = pSG->GetConnectedMediaType( &mt );
        if ( FAILED( hr) )
        {
            MSGBOX(NULL, TEXT("Could not read the connected media type"));
            return hr;
        }

        VIDEOINFOHEADER * vih = (VIDEOINFOHEADER*) mt.pbFormat;
        nWidth = vih->bmiHeader.biWidth;// = 320;
        nHeight = vih->bmiHeader.biHeight;// = 240;

        // don't buffer the samples as they pass through
        //
        hr = pSG->SetBufferSamples( FALSE );
        if( FAILED( hr )){
            MSGBOX( NULL, "SetBufferSamples()" );
            return hr;
        }

        // only grab one at a time, stop stream after
        // grabbing one sample
        //
        pSG->SetOneShot( FALSE );
        if( FAILED( hr )){
            MSGBOX( NULL, "SetOneShot()" );
            return hr;
        }

        // set the callback, so we can grab the one sample
        //
```

```
pSG->SetCallback( &CallBack, 1 );
if( FAILED( hr )){
    MSGBOX( NULL, "SetCallBack()" );
    return hr;
}


// QueryInterface for DirectShow interfaces
hr = pGB->QueryInterface(IID_IMediaControl, (void **)&pMC);
if( FAILED( hr )){
    MSGBOX( NULL, "MediaControl" );
    return hr;
}
hr = pGB->QueryInterface(IID_IMediaEventEx, (void **)&pME);
if( FAILED( hr )){
    MSGBOX( NULL, "MediaEventEx" );
    return hr;
}
hr = pGB->QueryInterface(IID_IMediaSeeking, (void **)&pMS);
if( FAILED( hr )){
    MSGBOX( NULL, "MediaSeeking" );
    return hr;
}
hr = pGB->QueryInterface(IID_IMediaPosition, (void **)&pMP);
if( FAILED( hr )){
    MSGBOX( NULL, "MediaPosition" );
    return hr;
}

// Query for video interfaces, which may not be relevant for audio files
hr = pGB->QueryInterface(IID_IVideoWindow, (void **)&pVW);
if( FAILED( hr )){
    MSGBOX( NULL, "VideoWindow" );
    return hr;
}
hr = pGB->QueryInterface(IID_IBasicVideo, (void **)&pBV);
if( FAILED( hr )){
    MSGBOX( NULL, "BasicVideo" );
    return hr;
}


// Query for audio interfaces, which may not be relevant for video-only files
hr = pGB->QueryInterface(IID_IBasicAudio, (void **)&pBA);
if( FAILED( hr )){
    MSGBOX( NULL, "BasicAudio" );
    return hr;
}

// ボリュームは0
pBA->put_Volume(-10000L);

// Have the graph signal event via window callbacks for performance
```



```
        hr = pME->SetNotifyWindow((OAHWND)m_hWnd, WM_GRAPHNOTIFY, 0);
        if( FAILED( hr )){
            MSGBOX( NULL, "pME->SetNotifyWindow" );
            return hr;
        }
        hr = pVW->put_Owner((OAHWND)m_hWnd);
        if( FAILED( hr )){
            MSGBOX( NULL, "pVW->put_Owner" );
            return hr;
        }
        hr = pVW->put_WindowStyle( WS_CLIPSIBLINGS | WS_CLIPCHILDREN|WS_VISIBLE);
        if( FAILED( hr )){
            MSGBOX( NULL, "pVW->put_WindowStyle" );
            return hr;
        }
//      InitVideoWindow(1, 1);
        MoveVideoWindow();
    //  GetFrameStepInterface();

        ShowWindow( SW_SHOWNORMAL );
        UpdateWindow();
        SetForegroundWindow();
        SetFocus();



#ifdef REGISTER_FILTERGRAPH
        hr = AddGraphToRot(pGB, &g_dwGraphRegister);
        if (FAILED(hr))
        {
            MSGBOX(NULL, TEXT("Failed to register filter graph with ROT!  hr=0x%x"),
hr);
            g_dwGraphRegister = 0;
        }
#endif

    }
}

HRESULT CMediaPlayWnd::InitVideoWindow(int nMultiplier, int nDivider)
{
    LONG lHeight, lWidth;
    HRESULT hr = S_OK;
    RECT rect;

    if (!pBV)
        return S_OK;

    // Read the default video size
    hr = pBV->GetVideoSize(&lWidth, &lHeight);
    if (hr == E_NOINTERFACE)
```

```
        return S_OK;

    // Account for requests of normal, half, or double size
    lWidth  = lWidth  * nMultiplier / nDivider;
    lHeight = lHeight * nMultiplier / nDivider;

    ::SetWindowPos(m_hWnd, NULL, 0, 0, lWidth, lHeight,
                SWP_NOMOVE | SWP_NOOWNERZORDER);

    UpdateWindow();
/*
    int nTitleHeight  = 0;//GetSystemMetrics(SM_CYCAPTION);
    int nBorderWidth  = 0;//GetSystemMetrics(SM_CXBORDER);
    int nBorderHeight = 0;//GetSystemMetrics(SM_CYBORDER);

    // Account for size of title bar and borders for exact match
    // of window client area to default video size
    /::SetWindowPos(m_hWnd, NULL, 0, 0, lWidth + 2*nBorderWidth,
            lHeight + nTitleHeight + 2*nBorderHeight,
            SWP_NOMOVE | SWP_NOOWNERZORDER);

    GetClientRect( &rect);

    APITRACE("%d,%d\n",rect.top, rect.left);
    hr = pVW->SetWindowPosition(rect.left, rect.top, rect.right/2, rect.bottom/2);

    if( FAILED(hr)){
        APITRACE("Failed  \n");
    }
*/
     return hr;
}

void CMediaPlayWnd::Play(void)
{
    if (!pMC)
        return;
    MoveVideoWindow();
    pMC->Run();
    pSG->SetCallback( &CallBack, 1 );
    SetFocus();
}

void CMediaPlayWnd::Pause(void)
{
    if (!pMC)
        return;
    MoveVideoWindow();
    pMC->Pause();

}

void CMediaPlayWnd::Stop(void)
```



```
{
    HRESULT hr;

    if ((!pMC) || (!pMS))
        return;

    if( pSG )
        pSG->SetCallback( NULL, 1 );

    // Stop and reset postion to beginning
    LONGLONG pos = 0;
    hr = pMC->Stop();

    // Seek to the beginning
    hr = pMS->SetPositions(&pos, AM_SEEKING_AbsolutePositioning ,
         NULL, AM_SEEKING_NoPositioning);

    // Display the first frame to indicate the reset condition
    hr = pMC->Pause();


    MoveVideoWindow();
}

void CMediaPlayWnd::MoveVideoWindow(void)
{
    HRESULT hr;

    // Track the movement of the container window and resize as needed
    if(pVW)
    {
        RECT client;

        GetClientRect(&client);
        hr = pVW->SetWindowPosition(client.left, client.top,
            client.right-client.left, client.bottom- client.top);
        UpdateWindow();
    }
}

void CMediaPlayWnd::OnSize(UINT nType, int cx, int cy)
{
    MoveVideoWindow();

    CWnd::OnSize(nType, cx, cy);

}
void CMediaPlayWnd::OnMove(int x, int y)
{
    MoveVideoWindow();
    CWnd::OnMove(x, y);
}
BOOL CMediaPlayWnd::GetFrameStepInterface(void)
```

```
{
    HRESULT hr;
    IVideoFrameStep *pFSTest = NULL;

    // Get the frame step interface, if supported
    hr = pGB->QueryInterface(__uuidof(IVideoFrameStep), (PVOID *)&pFSTest);
    if (FAILED(hr))
        return FALSE;

    // Check if this decoder can step
    hr = pFSTest->CanStep(0L, NULL);

    if (hr == S_OK)
    {
        pFS = pFSTest;  // Save interface to global variable for later use
        return TRUE;
    }
    else
    {
        pFSTest->Release();
        return FALSE;
    }
}


HRESULT CMediaPlayWnd::ModifyRate(double  dRateAdjust)
{
    HRESULT hr=S_OK;
    double dRate;

    // If the IMediaPosition interface exists, use it to set rate
    if ((pMP) && (dRateAdjust != 0))
    {
        if ((hr = pMP->get_Rate(&dRate)) == S_OK)
        {
            // Add current rate to adjustment value
            double dNewRate = dRate + dRateAdjust;
            hr = pMP->put_Rate(dNewRate);
        }
    }
    if( FAILED(hr)){
        APITRACE("FAILED ModifyRate hr=%d", hr);
    }
    return hr;
}

HRESULT CMediaPlayWnd::SetRate(double dRate)
{
    HRESULT hr=S_OK;

    if( pSG ){
        hr = pSG->SetCallback(NULL,1);
    }
```



```
    // If the IMediaPosition interface exists, use it to set rate
    if (pMP)
    {
        hr = pMP->put_Rate(dRate);
    }

    if( pSG ){
        hr = pSG->SetCallback( &CallBack,1);
    }
    if( FAILED(hr)){
        APITRACE("FAILED SetRate hr=%d", hr);
    }

    return hr;
}

HRESULT CMediaPlayWnd::GetDuration(REFTIME *pDuration)
{
    *pDuration = 0;
    if( pMP )
        pMP->get_Duration( pDuration );
    return E_NOTIMPL;
}

HRESULT CMediaPlayWnd::SetPosPercent(int percent)
{
    pSG->SetCallback( NULL, 1 );


    HRESULT hr;
    REFTIME pos = 0;
    REFTIME duration;
    pMP->get_Duration( &duration );

    pos = duration * percent / 100.0;
    hr = pMP->put_CurrentPosition( pos );



    pSG->SetCallback( &CallBack, 1 );
    return E_NOTIMPL;
}

void CMediaPlayWnd::OnPaint()
{
    CPaintDC dc(this); // device context for painting
    MoveVideoWindow();
}
```

```
//-----------------------------------------------------------------------------
//
//  スナップショットダイアログ ヘッダ
//
//      DlgSnapshot.h
//
//  VC++.NET 2003                       2004/11/01  THas
//
//-----------------------------------------------------------------------------
#pragma once
//-----------------------------------------------------------------------------
//  includes
//-----------------------------------------------------------------------------

#include "CMediaPlayWnd.h"


//-----------------------------------------------------------------------------
//  definition
//-----------------------------------------------------------------------------

#ifndef WM_USER_SNAPSHOT
#define WM_USER_SNAPSHOT    WM_USER + 100
    //  wParam : フレームバッファのサイズ
    //  lParam : フレームバッファのポインタ
#endif


//-----------------------------------------------------------------------------
//  CDlgDeviceSelect class
//-----------------------------------------------------------------------------

// CDlgSnapshot ダイアログ

class CSnapshot : public CDialog
{
    DECLARE_DYNCREATE(CSnapshot)

public:
    CSnapshot(CWnd* pParent = NULL);   // 標準コンストラクタ
    ~CSnapshot();
    BOOL SetFrameData(long nSize, BYTE *pData);
    BOOL AllocFrameBuff(long nSize);

protected:
    //  variables
    ULONG               m_nBitmapInfoSize;
    BITMAPINFO          *m_pBitmapInfo;
    long                m_nFrameBuffSize;
    BYTE                *m_pFrameBuff;

    virtual BOOL OnInitDialog();
```



```
private:

    //  helper function
    BOOL DrawFrame(BYTE *pData);
}
```

```cpp
#include "stdafx.h"
#include "Snapshot.h"
#include <CDIB.h>


CSnapshot::CSnapshot(CWnd* pParent /*=NULL*/)
//	: CDialog()
{
	//	モジュール変数初期化
	m_nBitmapInfoSize = 0;
	m_pBitmapInfo     = NULL;
	m_nFrameBuffSize  = 0;
	m_pFrameBuff      = NULL;
}

CSnapshot::~CSnapshot()
{
	//	モジュール変数クリア
	if(NULL != m_pBitmapInfo) {
		delete[] m_pBitmapInfo;
		//m_pBitmapInfo = NULL;
	}
	//m_nBitmapInfoSize = 0;
	if(NULL != m_pFrameBuff) {
		delete[] m_pFrameBuff;
		//m_pFrameBuff = NULL;
	}
	//m_nFrameBuffSize  = 0;
}

//	フレームバッファを割り当てる
//---------------------------------------------------------------------
//
//	引数
//		nSize : 割り当てるバッファサイズ(バイトサイズ)
//
//	戻り値
//		成功すれば TRUE
//		失敗すれば FALSE を返す
//
BOOL CSnapshot::AllocFrameBuff(long nSize)
{
	//	fool trap
	if(nSize < 0)	return FALSE;

	//	既に確保されている場合は何もしない
	if(nSize == m_nFrameBuffSize) {
		//memset(m_pFrameBuff, 0x00, m_FrameBuffSize);
		return TRUE;
	}

	//	現在確保中の領域を解放する
	if(nSize != m_nFrameBuffSize) {
```



```cpp
		if(NULL != m_pFrameBuff) {
			delete[] m_pFrameBuff;
			m_pFrameBuff = NULL;
		}
		m_nFrameBuffSize = 0;
	}

	//	新しい領域を確保する
	if(0 < nSize) {
		m_pFrameBuff = new BYTE[nSize];
		if(NULL == m_pFrameBuff) {
			return FALSE;
		}
		m_nFrameBuffSize = nSize;
		memset(m_pFrameBuff, 0x00, m_nFrameBuffSize);
	}

	return TRUE;
}

//	フレームデータを設定する
//---------------------------------------------------------------------
//
//	引数
//		nSize : データのサイズ(バイトサイズ)
//		pData : 画像データのポインタ
//
//	戻り値
//		成功すれば TRUE
//		失敗すれば FALSE を返す
//
BOOL CSnapshot::SetFrameData(long nSize, BYTE *pData)
{
	//	fool trap
	if(nSize < 0)		return FALSE;
	if(NULL == pData)	return FALSE;

	//	必要なら保存領域を確保する
	if(m_nFrameBuffSize < nSize) {
		AllocFrameBuff(nSize);
		if(nSize != m_nFrameBuffSize)	return FALSE;
	}

	//	データをコピーする
	if(0 < nSize) {
		memcpy(m_pFrameBuff, pData, nSize);
	}

	//	画面を描画する
	DrawFrame(m_pFrameBuff);

	return TRUE;
}
```

```
//-----------------------------------------------------------------------------
//  helper functions
//-----------------------------------------------------------------------------

//  フレーム(画像)を描画する
//-----------------------------------------------------------------
//
//  引数
//      pData : 画像データのポインタ
//
//  戻り値
//      成功すれば TRUE
//      失敗すれば FALSE を返す
//
BOOL CSnapshot::DrawFrame(BYTE *pData)
{
    //  fool trap
    if(NULL == pData)          return FALSE;
    if(NULL == m_pBitmapInfo)  return FALSE;

    //  draw frame
    BITMAPINFOHEADER *pbmh = reinterpret_cast<BITMAPINFOHEADER*>(m_pBitmapInfo);
    CDC              *pcdc = GetDC();
    SetDIBitsToDevice(pcdc->m_hDC, 0, 0,
                      pbmh->biWidth, pbmh->biHeight, 0, 0,
                      0, pbmh->biHeight,
                      pData,
                      m_pBitmapInfo,
                      DIB_RGB_COLORS);
    ReleaseDC(pcdc);

    return TRUE;
}

//-----------------------------------------------------------------------------
//  events (override)
//-----------------------------------------------------------------------------

//  ダイアログ初期化
BOOL CSnapshot::OnInitDialog()
{
    CDialog::OnInitDialog();

    return TRUE;
}
```

```
// stdafx.h : 標準のシステム インクルード ファイルのインクルード ファイル、または
// 参照回数が多く、かつあまり変更されない、プロジェクト専用のインクルード ファイル
// を記述します。

#pragma once

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN        // Windows ヘッダーから使用されていない部分を除外します。
#endif

// 下で指定された定義の前に対象プラットフォームを指定しなければならない場合、以下の定義を変更してください。
// 異なるプラットフォームに対応する値に関する最新情報については、MSDN を参照してください。
#ifndef WINVER              // Windows 95 および Windows NT 4 以降のバージョンに固有の機能の使用を許可します。
#define WINVER 0x0400       // これを Windows 98 および Windows 2000 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_WINNT        // Windows NT 4 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINNT 0x0400     // これを Windows 98 および Windows 2000 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_WINDOWS      // Windows 98 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINDOWS 0x0410 // これを Windows Me またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_IE           // IE 4.0 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_IE 0x0400    // これを IE 5.0  またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS  // 一部の CString コンストラクタは明示的です。

// 一般的で無視しても安全な MFC の警告メッセージの一部の非表示を解除します。
#define _AFX_ALL_WARNINGS

#include <afxwin.h>         // MFC のコアおよび標準コンポーネント
#include <afxext.h>         // MFC の拡張部分
#include <afxdisp.h>        // MFC オートメーション クラス

#include <afxdtctl.h>       // MFC の Internet Explorer 4 コモン コントロール サポート
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>         // MFC の Windows コモン コントロール サポート
#endif // _AFX_NO_AFXCMN_SUPPORT
#include <windows.h>

#ifndef _AFX_NOFORCE_LIBS

//////////////////////////////////////////////////////////////////////////////
// Win32 libraries
```



```
#ifndef _AFXDLL
    #ifndef _UNICODE
        #ifdef _DEBUG
            #pragma comment(lib, "nafxcwd.lib")
        #else
            #pragma comment(lib, "nafxcw.lib")
        #endif
    #else
        #ifdef _DEBUG
            #pragma comment(lib, "uafxcwd.lib")
        #else
            #pragma comment(lib, "uafxcw.lib")
        #endif
    #endif
#else
    #ifndef _UNICODE
        #ifdef _DEBUG
            #pragma comment(lib, "mfc71d.lib")
            #pragma comment(lib, "mfcs71d.lib")
        #else
            #pragma comment(lib, "mfc71.lib")
            #pragma comment(lib, "mfcs71.lib")
        #endif
    #else
        #ifdef _DEBUG
            #pragma comment(lib, "mfc71ud.lib")
            #pragma comment(lib, "mfcs71ud.lib")
        #else
            #pragma comment(lib, "mfc71u.lib")
            #pragma comment(lib, "mfcs71u.lib")
        #endif
    #endif
#endif

#ifdef _DLL
    #if !defined(_AFX_NO_DEBUG_CRT) && defined(_DEBUG)
        #pragma comment(lib, "msvcrtd.lib")
    #else
        #pragma comment(lib, "msvcrt.lib")
    #endif
#else
#ifdef _MT
    #if !defined(_AFX_NO_DEBUG_CRT) && defined(_DEBUG)
        #pragma comment(lib, "libcmtd.lib")
    #else
        #pragma comment(lib, "libcmt.lib")
    #endif
#else
    #if !defined(_AFX_NO_DEBUG_CRT) && defined(_DEBUG)
        #pragma comment(lib, "libcd.lib")
    #else
        #pragma comment(lib, "libc.lib")
    #endif
```

```
#endif
#endif
#pragma comment(lib, "libcmt.lib")
#pragma comment(lib, "kernel32.lib")
#pragma comment(lib, "user32.lib")
#pragma comment(lib, "gdi32.lib")
#pragma comment(lib, "msimg32.lib")
#pragma comment(lib, "comdlg32.lib")
#pragma comment(lib, "winspool.lib")
#pragma comment(lib, "advapi32.lib")
#pragma comment(lib, "shell32.lib")
#pragma comment(lib, "comctl32.lib")
#pragma comment(lib, "shlwapi.lib")

// force inclusion of NOLIB.OBJ for /disallowlib directives
#pragma comment(linker, "/include:__afxForceEXCLUDE")

// force inclusion of DLLMODUL.OBJ for _USRDLL
#ifdef _USRDLL
#pragma comment(linker, "/include:__afxForceUSRDLL")
#endif

// force inclusion of STDAFX.OBJ for precompiled types
#ifdef _AFXDLL
#pragma comment(linker, "/include:__afxForceSTDAFX")
#endif

#endif //!_AFX_NOFORCE_LIBS
```

```
// stdafx.cpp : 標準インクルードを含むソース ファイルです。
// VideoEffector.pch は、プリコンパイル済みヘッダーになります。
// stdafx.obj にはプリコンパイル済み方情報が含まれます。

#include "stdafx.h"
```

```
//-------------------------------------------------------------------------------
//
//  DirectShow ユーティリティヘッダ
//
//      UtilDShow.h
//
//  VC++.NET 2003               2004/11/01  THos
//
//-------------------------------------------------------------------------------
#ifndef __UtilDShow_h__
#define __UtilDShow_h__
//-------------------------------------------------------------------------------
//  include files
//-------------------------------------------------------------------------------

#include <windows.h>
#include <dshow.h>


//-------------------------------------------------------------------------------
//  definitions
//-------------------------------------------------------------------------------


//-------------------------------------------------------------------------------
//  prototypes
//-------------------------------------------------------------------------------

HRESULT DS_EnumVideoInputDevices(IMoniker *pMonikerList[], const int nListSize, int
*pEnumCount);
HRESULT DS_EnumVideoInputDevices(const HWND hLst);
HRESULT DS_GetVideoInputFilter(LPCTSTR pszFilterName, IBaseFilter **ppFilter);

HRESULT DS_EnumDirectShowFilters(IMoniker *pMonikerList[], const int nListSize, int
*pEnumCount);
HRESULT DS_EnumDirectShowFilters(const HWND hLst);
HRESULT DS_GetDirectShowFilter(LPCTSTR pszFilterName, IBaseFilter **ppFilter);


HRESULT DS_EnumPins(const HWND hLst, IBaseFilter *pFilter);
HRESULT DS_GetPin(IBaseFilter *pFilter, LPCTSTR pszPinName, const PIN_DIRECTION
PinDir, IPin **ppPin);
HRESULT DS_GetPinByCategory(IBaseFilter *pFilter, const GUID& Category, const
PIN_DIRECTION PinDir, IPin **ppPin);



void DS_FreeMediaType(AM_MEDIA_TYPE *pmt);
AM_MEDIA_TYPE *DS_CopyMediaType(AM_MEDIA_TYPE *pDst, AM_MEDIA_TYPE const *pSrc);

void DS_ShowErrMsg(HWND hWnd, HRESULT hr);
```



```
//-------------------------------------------------------------------------------
#endif
```

```
//-----------------------------------------------------------------------------
//
//  DirectShow ユーティリティモジュール
//
//      UtilDShow.cpp
//
//  VC++.NET 2003               2004/11/01  THas
//
//-----------------------------------------------------------------------------
#include "UtilDShow.h"
//-----------------------------------------------------------------------------
//  include files
//-----------------------------------------------------------------------------

#include <atlbase.h>


//-----------------------------------------------------------------------------
//  prototypes
//-----------------------------------------------------------------------------

static HRESULT DS_EnumFilters(REFCLSID ClassID, IMoniker *pMonikerList[], const int
nListSize, int *pEnumCount);
static HRESULT DS_EnumFilters(REFCLSID ClassID, const HWND hLst);
static HRESULT DS_GetFilter(REFCLSID ClassID, LPCTSTR pszFilterName, IBaseFilter
**ppFilter);

static BOOL DS_MatchesPinCategory(IPin *pPin, const GUID& Category);


//-----------------------------------------------------------------------------
//  public functions
//-----------------------------------------------------------------------------


//-----------------------------------------------------------------------------
//  フィルタ関連
//-----------------------------------------------------------------------------

//  ビデオキャプチャデバイスを列挙する
//-------------------------------------------------------------
//  ビデオキャプチャフィルタのデバイスモニカを列挙する
//
//  引数
//      pMonikerList : 取得したデバイスモニカを返す
//      nListSize    : pMonikerList の配列要素数
//      pEnumCount   : 取得したデバイスモニカの数を返す
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_EnumVideoInputDevices(IMoniker *pMonikerList[], const int nListSize, int
*pEnumCount)
```



```
{
    HRESULT hr;
    hr = DS_EnumFilters(CLSID_VideoInputDeviceCategory,
                        pMonikerList, nListSize, pEnumCount);
    return hr;
}


//  ビデオキャプチャデバイスを列挙する
//-------------------------------------------------------------
//  ビデオキャプチャフィルタの FriendlyName を hLst に列挙する
//
//  引数
//      hLst : 表示先のリストボックスのウインドウハンドル
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_EnumVideoInputDevices(const HWND hLst)
{
    HRESULT hr;
    hr = DS_EnumFilters(CLSID_VideoInputDeviceCategory, hLst);
    return hr;
}


//  指定のビデオキャプチャフィルタを取得する
//-------------------------------------------------------------
//  pszFilterName のビデオキャプチャフィルタを取得する
//
//  引数
//      pszFilterName : 取得するフィルタの名前(FriendlyName)
//      ppFilter      : 取得したフィルタを返す
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_GetVideoInputFilter(LPCTSTR pszFilterName, IBaseFilter **ppFilter)
{
    HRESULT     hr;
    hr = DS_GetFilter(CLSID_VideoInputDeviceCategory, pszFilterName, ppFilter);
    return hr;
}


//  DirectShow フィルタを列挙する
//-------------------------------------------------------------
//  DirectShow フィルタのデバイスモニカを列挙する
//
//  引数
//      pMonikerList : 取得したデバイスモニカを返す
//      nListSize    : pMonikerList の配列要素数
//      pEnumCount   : 取得したデバイスモニカの数を返す
```

```
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_EnumDirectShowFilters(IMoniker *pMonikerList[], const int nListSize, int
*pEnumCount)
{
    HRESULT hr;
    hr = DS_EnumFilters(CLSID_LegacyAmFilterCategory,
                        pMonikerList, nListSize, pEnumCount);
    return hr;
}


//  DirectShow フィルタを列挙する
//-------------------------------------------------------------------
//  DirectShow フィルタの FriendlyName を hLst に列挙する
//
//  引数
//      hLst : 表示先のリストボックスのウインドウハンドル
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_EnumDirectShowFilters(const HWND hLst)
{
    HRESULT hr;
    hr = DS_EnumFilters(CLSID_LegacyAmFilterCategory, hLst);
    return hr;
}


//  DirectShow フィルタを取得する
//-------------------------------------------------------------------
//  pszFilterName の DirectShow フィルタを取得する
//
//  引数
//      pszFilterName : 取得するフィルタの名前(FriendlyName)
//      ppFilter      : 取得したフィルタを返す
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_GetDirectShowFilter(LPCTSTR pszFilterName, IBaseFilter **ppFilter)
{
    HRESULT     hr;
    hr = DS_GetFilter(CLSID_LegacyAmFilterCategory, pszFilterName, ppFilter);
    return hr;
}


//-----------------------------------------------------------------------------
//  ピン関連
```



```
//-----------------------------------------------------------------------------

//  指定のフィルタのピンを列挙する
//-------------------------------------------------------------------
//  pFilter のピン名を hLst に列挙する
//
//  引数
//      hLst    : 表示先のリストボックスのウインドウハンドル
//      pFilter : フィルタ(検索対象)
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_EnumPins(const HWND hLst, IBaseFilter *pFilter)
{

    HRESULT hr;

    //  fool trap
    if(NULL == hLst)     return E_POINTER;
    if(NULL == pFilter)  return E_POINTER;

    //  enumerate devices ready
    IEnumPins   *pEnum;
    IPin        *pPin;

    hr = pFilter->EnumPins(&pEnum);
    if(S_OK != hr) {
        return hr;
    }

    //  enumerate devices
    while(S_OK == pEnum->Next(1, &pPin, NULL)) {

        PIN_INFO    PinInfo;

        memset(&PinInfo, 0x00, sizeof(PIN_INFO));
        hr = pPin->QueryPinInfo(&PinInfo);
        if(S_OK == hr) {
            USES_CONVERSION;
            SendMessage(hLst, LB_ADDSTRING, 0,
                        reinterpret_cast<LPARAM>(W2T(PinInfo.achName)));
            if(NULL != PinInfo.pFilter) {
                PinInfo.pFilter->Release();
            }
        }
        pPin->Release();
    }

    //  enumerate devices finalize
    pEnum->Release();

    return hr;
```

```
}


//  指定のフィルタのピンを取得する
//---------------------------------------------------------------
//  pFilter の pszPinName, PinDir のピンを取得する
//
//  引数
//      pFilter    : フィルタ           (検索対象)
//      pszPinName : 取得するピンの名前 (検索条件)
//      PinDir     : 取得するピンの方向 (検索条件)
//      ppPIn      : 取得したピンを返す
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_GetPin(IBaseFilter *pFilter, LPCTSTR pszPinName, const PIN_DIRECTION
PinDir, IPin **ppPin)
{
    HRESULT hr;

    //  fool trap
    if(NULL ==  pFilter)    return E_POINTER;
    if(NULL == pszPinName)  return E_POINTER;
    if(NULL == ppPin)       return E_POINTER;

    //  clear result
    *ppPin = NULL;

    //  enumerate devices ready
    IEnumPins   *pEnum;
    IPin        *pPin;

    hr = pFilter->EnumPins(&pEnum);
    if(S_OK != hr)  return hr;

    //  enumerate devices
    while(S_OK == pEnum->Next(1, &pPin, NULL)) {
        PIN_INFO    pin_info;
        hr = pPin->QueryPinInfo(&pin_info);
        if(S_OK == hr) {
            if(pin_info.dir == PinDir) {
                USES_CONVERSION;
                if(0 == lstrcmp(pszPinName, W2T(pin_info.achName))) {
                    *ppPin = pPin;
                    break;
                }
            }
        }
        pPin->Release();
    }

    //  enumerate devices finalize
```



```
    pEnum->Release();

    return hr;
}


//  指定のフィルタのピンを取得する
//---------------------------------------------------------------
//  pFilter の Category, PinDir のピンを取得する
//
//  引数
//      pFilte   : フィルタ              (検索対象)
//      Category : 取得するピンのカテゴリ (検索条件)
//      PinDir   : 取得するピンの方向    (検索条件)
//      ppPIn    : 取得したピンを返す
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_GetPinByCategory(IBaseFilter *pFilter, const GUID& Category, const
PIN_DIRECTION PinDir, IPin **ppPin)
{
    HRESULT hr;

    //  fool trap
    if(NULL == pFilter)  return E_POINTER;
    if(NULL == ppPin)    return E_POINTER;

    //  clear result
    *ppPin = NULL;

    //  enumerate devices ready
    IEnumPins   *pEnum;
    IPin        *pPin;

    hr = pFilter->EnumPins(&pEnum);
    if(S_OK != hr)  return hr;

    //  enumerate devices
    while(S_OK == pEnum->Next(1, &pPin, NULL)) {
        PIN_DIRECTION   pin_dir;
        hr = pPin->QueryDirection(&pin_dir);
        if(S_OK == hr) {
            if(DS_MatchesPinCategory(pPin, Category)) {
                *ppPin = pPin;
                break;
            }
        }
        pPin->Release();
    }

    //  enumerate devices finalize
    pEnum->Release();
```

```
    return hr;
}


//------------------------------------------------------------------------------
// ユーティリティ
//------------------------------------------------------------------------------

// AM_MEDIA_TYPE 構造体のリソースを解放する
//------------------------------------------------------------
//
// 引数
//      pmt : AM_MEDIA_TYPE 構造体のポインタ
//
// 戻り値
//      なし
//
void DS_FreeMediaType(AM_MEDIA_TYPE *pmt)
{
    // fool trap
    if(NULL == pmt)  return;

    // free pbFormat
    if(0 < pmt->cbFormat) {
        CoTaskMemFree(pmt->pbFormat);
        pmt->cbFormat = 0;
        pmt->pbFormat = NULL;
    }

    // free pUnk
    if(NULL != pmt->pUnk) {
        pmt->pUnk->Release();
        pmt->pUnk = NULL;
    }

    return;
}

// AM_MEDIA_TYPE 構造体をコピーする
//------------------------------------------------------------
//
// 引数
//      pDst : コピー先 AM_MEDIA_TYPE 構造体のポインタ
//      pSrc : コピー元 AM_MEDIA_TYPE 構造体のポインタ
//
// 戻り値
//      pDst を返す
//
AM_MEDIA_TYPE *DS_CopyMediaType(AM_MEDIA_TYPE *pDst, AM_MEDIA_TYPE const *pSrc)
{
    // fool trap
    if(NULL == pDst)  return NULL;
```



```
    if(NULL == pSrc) {
        // clear member
        memset(pDst, 0x00, sizeof(AM_MEDIA_TYPE));
    } else {
        // copy member
        memcpy(pDst, pSrc, sizeof(AM_MEDIA_TYPE));

        // not use pUnk
        pDst->pUnk = NULL;

        // allocate pbFormat
        if(0 < pDst->cbFormat) {
            pDst->pbFormat = reinterpret_cast<BYTE*>(CoTaskMemAlloc(pDst->cbFormat));
            if(NULL != pDst->pbFormat) {
                memcpy(pDst->pbFormat, pSrc->pbFormat, pDst->cbFormat);
            }
        } else {
            pDst->pbFormat = NULL;
        }
    }

    return pDst;
}

// エラーメッセージを表示する
//------------------------------------------------------------
//
// 引数
//      hWnd : メッセージダイアログの親ウインドウハンドル
//      hRes : エラーコード
//
// 戻り値
//      なし
//
void DS_ShowErrMsg(HWND hWnd, HRESULT hRes)
{
    if(FAILED(hRes))
    {
        TCHAR msg[MAX_ERROR_TEXT_LEN];
        if(0 == AMGetErrorText(hRes, msg, MAX_ERROR_TEXT_LEN))
        {
            wsprintf(msg, TEXT("未定義のエラー : 0x%2x"), hRes);
        }
        MessageBox(hWnd, msg, TEXT("エラー"), MB_ICONEXCLAMATION | MB_OK);
    }
    return;
}


//------------------------------------------------------------------------------
// local functions
//------------------------------------------------------------------------------
```

```
//  フィルタを列挙する
//---------------------------------------------------------------
//  ClassID のフィルタの デバイスモニカを列挙する
//
//  引数
//      ClassID      : デバイスカテゴリのクラス識別子
//      pMonikerList : 取得したデバイスモニカを返す
//      nListSize    : pMonikerList の配列要素数
//      pEnumCount   : 取得したデバイスモニカの数を返す
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_EnumFilters(REFCLSID ClassID, IMoniker *pMonikerList[], const int
nListSize, int *pEnumCount)
{
    HRESULT hr;

    //  fool trap
    if(NULL == pEnumCount)  return E_POINTER;

    //  clear counter
    *pEnumCount = 0;

    //  enumerate devices ready
    ICreateDevEnum  *pDevEnum = NULL;
    IEnumMoniker    *pEnum    = NULL;

    hr = CoCreateInstance(CLSID_SystemDeviceEnum, NULL,
                          CLSCTX_INPROC_SERVER, IID_ICreateDevEnum,
                          reinterpret_cast<LPVOID*>(&pDevEnum));
    if(S_OK != hr) {
        return hr;
    }
    hr = pDevEnum->CreateClassEnumerator(ClassID, &pEnum, 0);
    if(S_OK != hr) {
        pDevEnum->Release();
        return hr;
    }

    //  enumerate devices
    IMoniker    *pMoniker = NULL;
    while(S_OK == pEnum->Next(1, &pMoniker, NULL)) {
        if(NULL == pMonikerList) {
            pMoniker->Release();
            (*pEnumCount)++;
        } else {
            if((*pEnumCount) < nListSize) {
                pMonikerList[*pEnumCount] = pMoniker;
                (*pEnumCount)++;
            } else {
                pMoniker->Release();
                break;
```



```
            }
        }
    }

    //  enumerate devices finalize
    pEnum->Release();
    pDevEnum->Release();

    return hr;
}


//  フィルタを列挙する
//---------------------------------------------------------------
//  ClassID のフィルタの FriendlyName を hLst に列挙する
//
//  引数
//      ClassID : デバイスカテゴリのクラス識別子
//      hLst    : 表示先のリストボックスのウインドウハンドル
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_EnumFilters(REFCLSID ClassID, const HWND hLst)
{
    HRESULT hr;

    //  fool trap
    if(NULL == hLst)  return E_POINTER;

    //  enumerate devices ready
    ICreateDevEnum  *pDevEnum = NULL;
    IEnumMoniker    *pEnum    = NULL;

    hr = CoCreateInstance(CLSID_SystemDeviceEnum, NULL,
                          CLSCTX_INPROC_SERVER, IID_ICreateDevEnum,
                          reinterpret_cast<LPVOID*>(&pDevEnum));
    if(S_OK != hr) {
        return hr;
    }
    hr = pDevEnum->CreateClassEnumerator(ClassID, &pEnum, 0);
    if(S_OK != hr) {
        pDevEnum->Release();
        return hr;
    }

    //  enumerate devices
    IMoniker    *pMoniker = NULL;
    while(S_OK == pEnum->Next(1, &pMoniker, NULL)) {

        IPropertyBag    *pPropBag;
        hr = pMoniker->BindToStorage(0, 0, IID_IPropertyBag,
                                     reinterpret_cast<LPVOID*>(&pPropBag));
```

```
        if(S_OK != hr) {
            pMoniker->Release();
            continue;
        }

        VARIANT vName;
        VariantInit(&vName);
        hr = pPropBag->Read(L"FriendlyName", &vName, 0);
        if(S_OK == hr) {
            USES_CONVERSION;
            SendMessage(hLst, LB_ADDSTRING, 0,
                        reinterpret_cast<LPARAM>(OLE2T(vName.bstrVal)));
            VariantClear(&vName);
        } else {
            break;
        }

        pPropBag->Release();
        pMoniker->Release();
    }

    //  enumerate devices finalize
    pEnum->Release();
    pDevEnum->Release();

    return hr;
}


//  指定のフィルタを取得する
//-----------------------------------------------------------------
//  ClassID, pszFilterName に合致するフィルタを取得する
//
//  引数
//      ClassID       : デバイスカテゴリのクラス識別子
//      pszFilterName : 取得するフィルタの名前(FriendlyName)
//      ppFilter      : 取得したフィルタを返す
//
//  戻り値
//      エラーコードを返す
//
HRESULT DS_GetFilter(REFCLSID ClassID, LPCTSTR pszFilterName, IBaseFilter **ppFilter)
{
    HRESULT     hr;

    //  fool trap
    if(NULL ==  pszFilterName)  return E_POINTER;
    if(NULL ==  ppFilter)       return E_POINTER;

    //  clear result
    *ppFilter = NULL;

    //  enumerate devices ready
```



```
    ICreateDevEnum  *pDevEnum = NULL;
    IEnumMoniker    *pEnum    = NULL;

    hr = CoCreateInstance(CLSID_SystemDeviceEnum, NULL,
                          CLSCTX_INPROC_SERVER, IID_ICreateDevEnum,
                          reinterpret_cast<LPVOID*>(&pDevEnum));
    if(S_OK != hr) {
        return hr;
    }
    hr = pDevEnum->CreateClassEnumerator(ClassID, &pEnum, 0);
    if(S_OK != hr) {
        pDevEnum->Release();
        return hr;
    }

    //  search devices
    IMoniker    *pMoniker = NULL;
    BOOL        found = FALSE;
    while(S_OK == pEnum->Next(1, &pMoniker, NULL)) {

        IPropertyBag    *pPropBag;
        hr = pMoniker->BindToStorage(0, 0, IID_IPropertyBag,
                                     reinterpret_cast<LPVOID*>(&pPropBag));
        if(S_OK != hr) {
            pMoniker->Release();
            continue;
        }

        VARIANT vName;
        VariantInit(&vName);
        hr = pPropBag->Read(L"FriendlyName", &vName, 0);
        if(S_OK == hr) {
            USES_CONVERSION;
            found = (0 == lstrcmp(pszFilterName, OLE2T(vName.bstrVal)));
            VariantClear(&vName);
            if(found) {
                hr = pMoniker->BindToObject(NULL, NULL, IID_IBaseFilter,
                                            reinterpret_cast<LPVOID*>(ppFilter));
            }
        }

        pPropBag->Release();
        pMoniker->Release();
        if(found)  break;
    }

    //  enumerate devices finalize
    pEnum->Release();
    pDevEnum->Release();

    return hr;
}
```

```
//  指定のピンのカテゴリを調べる
//------------------------------------------------------------------
//  pPin のカテゴリが Category かどうか調べる
//
//  引数
//      pPin     : 調べるピンのポインタ
//      Category : 調べるピンのカテゴリ
//                  Cf. PIN_CATEGORY_PREVIEW
//                      PIN_CATEGORY_CAPTURE
//                      PIN_CATEGORY_CC
//  戻り値
//      pPin のカテゴリが Catrgory なら TRUE
//      そうでなければ                  FALSE を返す
//
BOOL DS_MatchesPinCategory(IPin *pPin, const GUID& Category)
{
    HRESULT hr;
    BOOL    found = FALSE;

    //  fool trap
    if(pPin == NULL)  return found;

    //  check pin category
    IKsPropertySet  *pKs;
    hr = pPin->QueryInterface(IID_IKsPropertySet, (void **)&pKs);
    if(NULL != pKs) {
        GUID    PinCategory;
        DWORD   cbReturned;
        hr = pKs->Get(AMPROPSETID_Pin, AMPROPERTY_PIN_CATEGORY, NULL, 0,
                      &PinCategory, sizeof(GUID), &cbReturned);
        if(S_OK == hr) {
            found= (PinCategory == Category);
        }
        pKs->Release();
    }

    return found;
}


//------------------------------------------------------------------------------
```

```
// VideoEffector.h : VideoEffector アプリケーションのメイン ヘッダー ファイル
//
#pragma once

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // メイン シンボル


// CVideoEffectorApp:
// このクラスの実装については、VideoEffector.cpp を参照してください。
//

class CVideoEffectorApp : public CWinApp
{
public:
    CVideoEffectorApp();
    ~CVideoEffectorApp();


// オーバーライド
public:
    virtual BOOL InitInstance();

// 実装
    afx_msg void OnAppAbout();
    DECLARE_MESSAGE_MAP()
};

extern CVideoEffectorApp theApp;
```

```
// VideoEffector.cpp : アプリケーションのクラス動作を定義します。
//

#include "stdafx.h"
#include "VideoEffector.h"
#include "MainFrm.h"

#include "VideoEffectorDoc.h"
#include "VideoEffectorView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CVideoEffectorApp

BEGIN_MESSAGE_MAP(CVideoEffectorApp, CWinApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
    // 標準のファイル基本ドキュメント コマンド
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
    // 標準の印刷セットアップ コマンド
    ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()


// CVideoEffectorApp コンストラクション

CVideoEffectorApp::CVideoEffectorApp()
{
    // TODO: この位置に構築用コードを追加してください。
    // ここに InitInstance 中の重要な初期化処理をすべて記述してください。
    //-------------------------------------------------------
    //  初期化処理
    //-------------------------------------------------------
    //  COM 利用開始
    if(S_OK != CoInitialize(NULL)) {
        MessageBox(NULL, TEXT("COMの初期化に失敗しました"),
                         TEXT("debug info"), MB_ICONSTOP | MB_OK);
    }
    //-------------------------------------------------------
}

CVideoEffectorApp::~CVideoEffectorApp()
{
    //-------------------------------------------------------
    //  終了処理
    //-------------------------------------------------------
    //  COM 利用停止
    CoUninitialize();
    //-------------------------------------------------------
}
```



```
// 唯一の CVideoEffectorApp オブジェクトです。

CVideoEffectorApp theApp;

// CVideoEffectorApp 初期化

BOOL CVideoEffectorApp::InitInstance()
{
    // アプリケーション マニフェストが visual スタイルを有効にするために、
    // ComCtl32.dll バージョン 6 以降の使用を指定する場合は、
    // Windows XP に InitCommonControls() が必要です。さもなければ、ウィンドウ作成はすべて失
敗します。
    InitCommonControls();

    CWinApp::InitInstance();


    // OLE ライブラリを初期化します。
    if (!AfxOleInit())
    {
        AfxMessageBox(IDP_OLE_INIT_FAILED);
        return FALSE;
    }
    AfxEnableControlContainer();
    // 標準初期化
    // これらの機能を使わずに、最終的な実行可能ファイルのサイズを縮小したい場合は、
    // 以下から、不要な初期化ルーチンを
    // 削除してください。
    // 設定が格納されているレジストリ キーを変更します。
    // TODO: この文字列を、会社名または組織名などの、
    // 適切な文字列に変更してください。
    SetRegistryKey(_T("アプリケーション ウィザードで生成されたローカル アプリケーション"));
    LoadStdProfileSettings(4);  // 標準の INI ファイルのオプションをロードします (MRU を含む)
    // アプリケーション用のドキュメント テンプレートを登録します。ドキュメント テンプレート
    //  はドキュメント、フレーム ウィンドウとビューを結合するために機能します。
    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CVideoEffectorDoc),
        RUNTIME_CLASS(CMainFrame),       // メイン SDI フレーム ウィンドウ
        RUNTIME_CLASS(CVideoEffectorView));
    AddDocTemplate(pDocTemplate);
    // DDE、file open など標準のシェル コマンドのコマンドラインを解析します。
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);
    // コマンド ラインで指定されたディスパッチ コマンドです。アプリケーションが
    // /RegServer、/Register、/Unregserver または /Unregister で起動された場合、 False を
返します。
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;
    // メイン ウィンドウが初期化されたので、表示と更新を行います。
```

```
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();
    // 接尾辞が存在する場合にのみ DragAcceptFiles を呼び出してください。
    //  SDI アプリケーションでは、ProcessShellCommand の直後にこの呼び出しが発生しなければなりま
せん。

    return TRUE;
}



// アプリケーションのバージョン情報に使われる CAboutDlg ダイアログ

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// ダイアログ データ
    enum { IDD = IDD_ABOUTBOX };

protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート

// 実装
protected:
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// ダイアログを実行するためのアプリケーション コマンド
void CVideoEffectorApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}


// CVideoEffectorApp メッセージ ハンドラ
```

```
// VideoEffectorDoc.h : CVideoEffectorDoc クラスのインターフェイス
//


#pragma once

class CVideoEffectorDoc : public CDocument
{
protected: // シリアル化からのみ作成します。
    CVideoEffectorDoc();
    DECLARE_DYNCREATE(CVideoEffectorDoc)

// 属性
public:

// 操作
public:

// オーバーライド
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);

// 実装
public:
    virtual ~CVideoEffectorDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// 生成された、メッセージ割り当て関数
protected:
    DECLARE_MESSAGE_MAP()
};
```

```
// VideoEffectorDoc.cpp : CVideoEffectorDoc クラスの実装
//

#include "stdafx.h"
#include "VideoEffector.h"

#include "VideoEffectorDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CVideoEffectorDoc

IMPLEMENT_DYNCREATE(CVideoEffectorDoc, CDocument)

BEGIN_MESSAGE_MAP(CVideoEffectorDoc, CDocument)
END_MESSAGE_MAP()


// CVideoEffectorDoc コンストラクション/デストラクション

CVideoEffectorDoc::CVideoEffectorDoc()
{
    // TODO: この位置に1度だけ呼ばれる構築用のコードを追加してください。

}

CVideoEffectorDoc::~CVideoEffectorDoc()
{
}

BOOL CVideoEffectorDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;

    // TODO: この位置に再初期化処理を追加してください。
    // (SDI ドキュメントはこのドキュメントを再利用します。)

    return TRUE;
}



// CVideoEffectorDoc シリアル化

void CVideoEffectorDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
```



```
        // TODO: 格納するコードをここに追加してください。
    }
    else
    {
        // TODO: 読み込むコードをここに追加してください。
    }
}


// CVideoEffectorDoc 診断

#ifdef _DEBUG
void CVideoEffectorDoc::AssertValid() const
{
    CDocument::AssertValid();
}

void CVideoEffectorDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG


// CVideoEffectorDoc コマンド
```

```
// VideoEffectorView.h : CVideoEffectorView クラスのインターフェイス
//


#pragma once
#include "ImageWnd.h"
#include "CMediaPlayWnd.h"



class CVideoEffectorView : public CView
{
protected: // シリアル化からのみ作成します。
    CVideoEffectorView();
    DECLARE_DYNCREATE(CVideoEffectorView)



// 属性
public:
    CVideoEffectorDoc* GetDocument() const;

    CImageWnd   ImageWnd;
    CMediaPlayWnd   *MediaPlayer;
    BITMAPINFO      *bmi;
    CSliderCtrl     Slider;

// 操作
public:

    int FilterNo;

// オーバーライド
    public:
    virtual void OnDraw(CDC* pDC);  // このビューを描画するためにオーバーライドされます。
//virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:

    TCHAR   m_szFileName[MAX_PATH];             //  ファイル名
    TCHAR   m_szDeviceName[MAX_FILTER_NAME];    //  カメラ名
    TCHAR   m_szOutPinName[MAX_PIN_NAME];       //  出力ピン名
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);

// 実装
public:
    virtual ~CVideoEffectorView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
```



```
protected:

// 生成された、メッセージ割り当て関数
protected:
    DECLARE_MESSAGE_MAP()
public:
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnFileOpen();
    afx_msg void OnMediaplayerPlay();
    afx_msg void OnMediaplayerPause();
    afx_msg void OnMediaplayerStop();
protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
public:
    int AddPlugins(void);
    void DoEffect(void);
    afx_msg void OnFileSave();
    afx_msg void OnFileSaveAs();
    afx_msg void OnMediaplayerDoubleplaybackrate();
    afx_msg void OnMediaplayerNormalplaybackrate();
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnFileClose();
    afx_msg void OnCameraOpen();
    afx_msg void OnPositionHead();
    afx_msg void OnPositionQuarter();
    afx_msg void OnPositionHalf();
    afx_msg void OnPositionThreequarter();

};

#ifndef _DEBUG  // VideoEffectorView.cpp のデバッグ バージョン
inline CVideoEffectorDoc* CVideoEffectorView::GetDocument() const
   { return reinterpret_cast<CVideoEffectorDoc*>(m_pDocument); }
#endif
```

```
// VideoEffectorView.cpp : CVideoEffectorView クラスの実装
//

#include "stdafx.h"
#include "VideoEffector.h"
#include <dshow.h>

#include "VideoEffectorDoc.h"
#include "VideoEffectorView.h"
#include "MainFrm.h"
#include "CMediaPlayWnd.h"
#include "CMediaPlayFile.h"
#include "CMediaPlayCamera.h"
#include "UtilDShow.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

extern CMainFrame *FrameWnd;

BOOL FileOpenDialog(LPTSTR pszFileName);

// CVideoEffectorView

IMPLEMENT_DYNCREATE(CVideoEffectorView, CView)

BEGIN_MESSAGE_MAP(CVideoEffectorView, CView)
    // 標準印刷コマンド
    ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
    ON_WM_CREATE()
    ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
    ON_COMMAND(ID_CAMERA_OPEN, OnCameraOpen)
    ON_COMMAND(ID_MEDIAPLAYER_PLAY, OnMediaplayerPlay)
    ON_COMMAND(ID_MEDIAPLAYER_PAUSE, OnMediaplayerPause)
    ON_COMMAND(ID_MEDIAPLAYER_STOP, OnMediaplayerStop)
    ON_COMMAND(ID_FILE_SAVE, OnFileSave)
    ON_COMMAND(ID_FILE_SAVE_AS, OnFileSaveAs)
    ON_COMMAND(ID_MEDIAPLAYER_DOUBLEPLAYBACKRATE, OnMediaplayerDoubleplaybackrate)
    ON_COMMAND(ID_MEDIAPLAYER_NORMALPLAYBACKRATE, OnMediaplayerNormalplaybackrate)
    ON_COMMAND(ID_HEAD, OnPositionHead)
    ON_COMMAND(ID_QUARTER, OnPositionQuarter)
    ON_COMMAND(ID_HALF, OnPositionHalf)
    ON_COMMAND(ID_THREEQUARTER, OnPositionThreequarter)
    ON_WM_TIMER()
    ON_WM_HSCROLL()
END_MESSAGE_MAP()

// CVideoEffectorView コンストラクション/デストラクション

CVideoEffectorView::CVideoEffectorView()
```



```
{
    // TODO: 構築コードをここに追加します。
    FilterNo = -1;
    lstrcpy(m_szDeviceName ,("Microsoft DV Camera and VCR"));
    lstrcpy(m_szOutPinName, ("DV A/V Out"));

    MediaPlayer = NULL;
    bmi = NULL;
}

CVideoEffectorView::~CVideoEffectorView()
{
    if(MediaPlayer != NULL)
        delete MediaPlayer;
    if(bmi != NULL)
        delete []bmi;

}

BOOL CVideoEffectorView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: この位置で CREATESTRUCT cs を修正して Window クラスまたはスタイルを
    //  修正してください。
;

    return CView::PreCreateWindow(cs);
}

// CVideoEffectorView 描画

void CVideoEffectorView::OnDraw(CDC* /*pDC*/)
{
    CVideoEffectorDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    // TODO: この場所にネイティブ データ用の描画コードを追加します。
}


// CVideoEffectorView 印刷

BOOL CVideoEffectorView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // デフォルトの印刷準備
    return DoPreparePrinting(pInfo);
}

void CVideoEffectorView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: 印刷前の特別な初期化処理を追加してください。
}

void CVideoEffectorView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
```

```
{
    // TODO: 印刷後の後処理を追加してください。
}


// CVideoEffectorView 診断

#ifdef _DEBUG
void CVideoEffectorView::AssertValid() const
{
    CView::AssertValid();
}

void CVideoEffectorView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CVideoEffectorDoc* CVideoEffectorView::GetDocument() const // デバッグ以外のバージョンはイ
ンラインです。
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CVideoEffectorDoc)));
    return (CVideoEffectorDoc*)m_pDocument;
}
#endif //_DEBUG


// CVideoEffectorView メッセージ ハンドラ
int CVideoEffectorView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{

    if (CView::OnCreate(lpCreateStruct) == -1)
        return -1;

    CString MyClass = AfxRegisterWndClass(NULL);

    RECT    rect;
    rect.top=0;
    rect.left=0;
    rect.right=100;
    rect.bottom=100;
    ImageWnd.Create( NULL, "ImageWnd",WS_BORDER|WS_CHILD |WS_VISIBLE,rect,this,0);
    ImageWnd.ShowWindow( SW_SHOW);

    rect.top = 240;
    rect.left = 320;
    rect.right = rect.left+320;
    rect.bottom = rect.top+30;

    Slider.Create(TBS_BOTH  ,rect,this,11111);
    return 0;
}
int BufferCBFunction( double dblSampleTime, BYTE * pBuffer, long lBufferSize ,void*
```



```
pData);
static int BufferCBFunction( double dblSampleTime, BYTE * pBuffer, long lBufferSize
,void* pData)
{

    CVideoEffectorView *pView = (CVideoEffectorView*)pData;

    pView->ImageWnd.DIB.CreateFromBitMapData( (pView->bmi), pBuffer);
    pView->DoEffect();

    pView->ImageWnd.MoveWindow( 0,0,pView->bmi->bmiHeader.biWidth,pView->bmi->bmiHead
er.biHeight);
    pView->ImageWnd.Invalidate();

    return 0;

}

void CVideoEffectorView::OnFileOpen()
{
    HRESULT hr;

    //  動作中であれば閉じる
    if(NULL != MediaPlayer) {
        OnFileClose();
    }

    //  ファイル名を入力させる
    if(!FileOpenDialog(m_szFileName))  return;

    //  ファイルを開く準備をする
    CMediaPlayFile  *pMPF;
    pMPF = new CMediaPlayFile(m_hWnd);
    if(NULL == pMPF) {
        MessageBox(TEXT("CMediaPlayFile を生成できませんでした"),
                   TEXT("エラー"),
                   MB_ICONEXCLAMATION | MB_OK);
        return;
    }

    //  ファイルを開く
    hr = pMPF->OpenClip(m_szFileName);
    if(S_OK != hr) {
        DS_ShowErrMsg(m_hWnd, hr);
    }

    MediaPlayer = pMPF;

    ULONG   bmi_size;
    MediaPlayer->GetBitmapInfo(&bmi_size, NULL);
    if(NULL != bmi) {
            delete[] reinterpret_cast<BYTE*>(bmi);
    }
```

```
    bmi = reinterpret_cast<BITMAPINFO*>(new BYTE[bmi_size]);
    MediaPlayer->GetBitmapInfo(&bmi_size, bmi);



    //  コールバックを設定する
    hr = MediaPlayer->SetBufferCB(&BufferCBFunction, static_cast<void*>(this));
    if(S_OK != hr) {
        DS_ShowErrMsg(m_hWnd, hr);
    }
    MediaPlayer->MoveWindow(MediaPlayer->GetBitmapWidth(),
0,MediaPlayer->GetBitmapWidth(),MediaPlayer->GetBitmapHeight());
    //  スタート
    hr = MediaPlayer->Play();

    if(S_OK != hr) {
        DS_ShowErrMsg(m_hWnd, hr);
    }

    MediaPlayer->Pause();


/*  MediaPlayer->m_SampleGrabberCB.SetBufferCBFunction
( (BUFFERCBFUNCTION)&BufferCBFunction,(void*)this);
    hr = MediaPlayer->Play();
    if(S_OK != hr) {
        DS_ShowErrMsg(m_hWnd, hr);
    }
    CFileDialog dlg(TRUE);
    if( dlg.DoModal() == IDOK ){

        MediaPlayer.OpenClip( dlg.GetPathName());

        bmi.bmiHeader.biSize=sizeof(BITMAPINFOHEADER); // BITMAPINFOHEADER 構造体
        bmi.bmiHeader.biWidth=MediaPlayer.nWidth;
        bmi.bmiHeader.biHeight=MediaPlayer.nHeight;
        bmi.bmiHeader.biPlanes=1;
        bmi.bmiHeader.biBitCount= 24;
        bmi.bmiHeader.biCompression=BI_RGB;
        bmi.bmiHeader.biSizeImage=0;
        bmi.bmiHeader.biXPelsPerMeter=0;
        bmi.bmiHeader.biYPelsPerMeter=0;
        bmi.bmiHeader.biClrUsed=0;
        bmi.bmiHeader.biClrImportant=0;

        MediaPlayer.MoveWindow( MediaPlayer.nWidth,0,MediaPlayer.nWidth,MediaPlayer.
nHeight);

        LONGLONG Duration;
        Slider.SetRange( 0, 100 );
        Slider.MoveWindow( MediaPlayer.nWidth,MediaPlayer.nHeight,MediaPlayer.
nWidth,30);
```



```
        //Slider.ShowWindow( SW_SHOW );

        MediaPlayer.Pause();
        MediaPlayer.MoveVideoWindow();
    }
*/
    Slider.SetRange( 0, 100 );
    Slider.MoveWindow
( MediaPlayer->GetBitmapWidth(),MediaPlayer->GetBitmapHeight(),MediaPlayer->GetBitmap
Width(),30);
}

void CVideoEffectorView::OnCameraOpen()
{
    HRESULT hr;

    //  動作中であれば閉じる
    if(NULL != MediaPlayer) {
        OnFileClose();
    }

    //  カメラを開く準備をする
    CMediaPlayCamera  *pMPC;
    pMPC = new CMediaPlayCamera(m_hWnd);
    if(NULL == pMPC) {
        MessageBox(TEXT("CMediaPlayCamera を生成できませんでした"),
                   TEXT("エラー"),
                   MB_ICONEXCLAMATION | MB_OK);
        return;
    }

    //  カメラを開く
    hr = pMPC->OpenClip(m_szDeviceName, m_szOutPinName);
    if(S_OK != hr) {
        DS_ShowErrMsg(m_hWnd, hr);
    }

    MediaPlayer = pMPC;

    ULONG   bmi_size;
    MediaPlayer->GetBitmapInfo(&bmi_size, NULL);
    if(NULL != bmi) {
            delete[] reinterpret_cast<BYTE*>(bmi);
    }

    bmi = reinterpret_cast<BITMAPINFO*>(new BYTE[bmi_size]);
    MediaPlayer->GetBitmapInfo(&bmi_size, bmi);



    //  コールバックを設定する
    hr = MediaPlayer->SetBufferCB(&BufferCBFunction, static_cast<void*>(this));
    if(S_OK != hr) {
```

```
        DS_ShowErrMsg(m_hWnd, hr);
    }
    MediaPlayer->MoveWindow(MediaPlayer->GetBitmapWidth(),
0,MediaPlayer->GetBitmapWidth(),MediaPlayer->GetBitmapHeight());
    //  スタート
    hr = MediaPlayer->Play();
    if(S_OK != hr) {
        DS_ShowErrMsg(m_hWnd, hr);
    }



/*  MediaPlayer->m_SampleGrabberCB.SetBufferCBFunction
( (BUFFERCBFUNCTION)&BufferCBFunction,(void*)this);
    hr = MediaPlayer->Play();
    if(S_OK != hr) {
        DS_ShowErrMsg(m_hWnd, hr);
    }
    CFileDialog dlg(TRUE);
    if( dlg.DoModal() == IDOK ){

        MediaPlayer.OpenClip( dlg.GetPathName());

        bmi.bmiHeader.biSize=sizeof(BITMAPINFOHEADER); // BITMAPINFOHEADER 構造体
        bmi.bmiHeader.biWidth=MediaPlayer.nWidth;
        bmi.bmiHeader.biHeight=MediaPlayer.nHeight;
        bmi.bmiHeader.biPlanes=1;
        bmi.bmiHeader.biBitCount= 24;
        bmi.bmiHeader.biCompression=BI_RGB;
        bmi.bmiHeader.biSizeImage=0;
        bmi.bmiHeader.biXPelsPerMeter=0;
        bmi.bmiHeader.biYPelsPerMeter=0;
        bmi.bmiHeader.biClrUsed=0;
        bmi.bmiHeader.biClrImportant=0;

        MediaPlayer.MoveWindow( MediaPlayer.nWidth,0,MediaPlayer.nWidth,MediaPlayer.
nHeight);

        LONGLONG Duration;
        Slider.SetRange( 0, 100 );
        Slider.MoveWindow( MediaPlayer.nWidth,MediaPlayer.nHeight,MediaPlayer.
nWidth,30);
        //Slider.ShowWindow( SW_SHOW );

        MediaPlayer.Pause();
        MediaPlayer.MoveVideoWindow();
    }
*/
    Slider.SetRange( 0, 100 );
    Slider.MoveWindow
( MediaPlayer->GetBitmapWidth(),MediaPlayer->GetBitmapHeight(),MediaPlayer->GetBitmap
Width(),30);
}
```



```
void CVideoEffectorView::OnMediaplayerPlay()
{
    MediaPlayer->Play();
}

void CVideoEffectorView::OnMediaplayerPause()
{
    MediaPlayer->Pause();
}

void CVideoEffectorView::OnMediaplayerStop()
{
    MediaPlayer->Stop();
}


void CVideoEffectorView::DoEffect(void)
{
    if( FilterNo != -1 ){
        FrameWnd->plgins[ FilterNo ]->DoFilter( ImageWnd.DIB.GetBitMapInfo(),ImageWnd
.DIB.GetImageData(),NULL,(long)this);
    }
}

void CVideoEffectorView::OnFileSave()
{
    CFileDialog dlg(FALSE);
    if( dlg.DoModal() == IDOK ){
        ImageWnd.DIB.SaveToFile( dlg.GetPathName() );
    }
}

void CVideoEffectorView::OnFileSaveAs()
{
    OnFileSave();
}

void CVideoEffectorView::OnMediaplayerDoubleplaybackrate()
{
    MediaPlayer->SetRate( 2.0 );
}

void CVideoEffectorView::OnMediaplayerNormalplaybackrate()
{
    MediaPlayer->SetRate( 1.0 );
}

void CVideoEffectorView::OnPositionHead()
{
    MediaPlayer->SetPosPercent(0.0);
}
void CVideoEffectorView::OnPositionQuarter()
```

```
{
    MediaPlayer->SetPosPercent(25.0);
}

void CVideoEffectorView::OnPositionHalf()
{
    MediaPlayer->SetPosPercent(50.0);
}

void CVideoEffectorView::OnPositionThreequarter()
{
    MediaPlayer->SetPosPercent(75.0);
}


void CVideoEffectorView::OnTimer(UINT nIDEvent)
{
    // TODO : ここにメッセージ ハンドラ コードを追加するか、既定の処理を呼び出します。

    CView::OnTimer(nIDEvent);
}

void CVideoEffectorView::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    MediaPlayer->SetPosPercent( nPos );

    //APITRACE("POS %d\n", nPos );

    CView::OnHScroll(nSBCode, nPos, pScrollBar);
}




//  ファイル・デバイス(カメラ)を閉じる
//-----------------------------------------------------------------
void CVideoEffectorView::OnFileClose()
{
    //  再生を停止してファイル・デバイス(カメラ)を閉じ解放する
    if(NULL != MediaPlayer) {
        MediaPlayer->Stop();
        MediaPlayer->CloseClip();
        delete MediaPlayer;
        MediaPlayer = NULL;

//        SetWindowText(m_szFormTitle);
    }
}
//-----------------------------------------------------------------------------
//  local functions
//-----------------------------------------------------------------------------

//  動画ファイルを選択する
```



```
//-----------------------------------------------------------------
//
//  引数
//      pszFileName : 選択したファイル名を返す(フルパス)
//
//  戻り値
//      ファイルが選択された場合 TRUE
//      選択されなかった場合     FALSE を返す
//
BOOL FileOpenDialog(LPTSTR pszFileName)
{
    static TCHAR buff[MAX_PATH];
    OPENFILENAME ofn;

    //  fool trap
    if(NULL == pszFileName)  return FALSE;

    memset(buff, 0x00, sizeof(TCHAR) * MAX_PATH);
    memset(&ofn, 0x00, sizeof(OPENFILENAME));

    ofn.lStructSize = sizeof(OPENFILENAME);
    ofn.hwndOwner         = NULL;
    ofn.hInstance         = NULL;
    ofn.lpstrFile         = TEXT("動画ファイル\0*.avi;*.asf;*.mpg;*.mpeg;*.mov;*.wmv\0
全てのファイル\0*.*\0\0");
    ofn.lpstrCustomFilter = NULL;
    ofn.nMaxCustFilter    = 0;
    ofn.nFilterIndex      = 1;
    ofn.lpstrFile         = buff;
    ofn.nMaxFile          = MAX_PATH;
    ofn.lpstrFileTitle    = NULL;
    ofn.nMaxFileTitle     = 0;
    ofn.lpstrInitialDir   = NULL;
    ofn.lpstrTitle        = TEXT("動画ファイルを開く");
    ofn.Flags             = OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;
    ofn.nFileOffset       = 0;
    ofn.nFileExtension    = 0;
    ofn.lpstrDefExt       = TEXT(".avi");
    ofn.lCustData         = 0;
    ofn.lpfnHook          = NULL;
    ofn.lpTemplateName    = NULL;

    if(GetOpenFileName(&ofn)) {
        lstrcpy(pszFileName, buff);
        return TRUE;
    } else {
        return FALSE;
    }
}
```