2008－CVIM－165 (26)
2008／11／28

ABMとMSCによる実時間移動物体検出

山本 文香†, 岩井 儀雄†, 谷内田 正彦‡
†大阪大学大学院基礎工学研究科システム創成専攻
〒560–8531 大阪府豊中市待兼山町 1–3
‡大阪工業大学情報科学部情報メディア学科
〒573–0196 大阪府枚方市北山 1–79–1
yamamoto@yachi-lab.sys.es.osaka-u.ac.jp, iwai@sys.es.osaka-u.ac.jp, yachida@is.oit.ac.jp

本稿では，GPU を利用することで移動物体領域の検出処理を高速化し，リアルタイムで高精度な移動物体検出を行うことを提案する．GPU(Graphics Processing Unit) は，全てのピクセルについて同じ計算を繰り返す処理を高速に実行できるので，GPU を利用することでピクセルベースの移動物体検出手法の高速化を図ることができる．GPU を利用して，ABM(Adaptive Background Model: 明るさ可変背景モデル) と MSC(Margined Sign Correlation : マージン付き符号相関) を用いた移動物体領域検出を行うことで，CPU を用いた場合より 9 倍以上高速に計算できることを確認した．このシステムを解像度 720×486 のカラー動画像に適用した結果とその処理時間を示す．

Real-Time Object Detection with Adaptive Background Model and Margined Sign Correlation

Ayaka YAMAMOTO†, Yoshio IWAI†, Masahiko YACHIDA‡
†Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka 560–8531 Japan
‡Faculty of Information Science and Technology, Osaka Institute of Technology
1–79–1 Kitayama, Hirakata, Osaka 573–0196 Japan
yamamoto@yachi-lab.sys.es.osaka-u.ac.jp, iwai@sys.es.osaka-u.ac.jp, yachida@is.oit.ac.jp

In this paper, we propose a fast and efficient method that detects moving objects. In order to achieve the fast performance, we use a GPU(Graphics Processing Unit). Since a GPU can execute the same calculation to each pixel at high speed, the object-detection method with the ABM(Adaptive Background Model) and the MSC (Margined Sign Correlation), a pixel-based method, can speed up by using a GPU. We show the experimental results using 720×486 color videos in order to verify that the object-detection process with ABM and MSC can be computed on the GPU over 9 times faster than the CPU.

1 はじめに

移動物体領域の検出は，人物の認識システムや交通状況の監視システム，セキュリティシステムなど，様々な分野で今後応用が期待される．画像中から移動物体を検出し，システムの自動化を図れば，監視作業の省力化や自動認識の実現が期待できる．このようなシステムへの応用を実現するためには，移動物体の検出処理を高精度にかつ実時間で行うことが望まれる．

これに対して，従来から多くの手法が提案されており，近年，移動物体の検出精度は大幅に向上している．なかでも，吉村らは明るさ可変背景モデル (Adaptive Background Model:ABM) とマージン付き符号相関 (Margined Sign Correlation:MSC) を用いた手法を提案しており，屋外環境において照明変動に対してロバストな検出を行い，かつ，移動物体とその影を正確に分離している [3]．しかし，提案システムはソフトウェアを用いており，CPU では実時間処理が困難であったためオフラインで移動物体検出をしていた．

一方で，高精度な移動物体の検出処理を高速化するために，専用ハードウェアによるシステムの

構築が試みられている [6, 7]. GPU(Graphics Processing Unit) は，近年の著しい性能の向上に伴い，GPGPU(General-Purpose computation on GPUs) と呼ばれる GPU により汎用的な数値計算を高速化する試みが注目を集めている [4, 5]. ABM と MSC を用いた手法は，全てのピクセルについて同じ計算を繰り返すので，これを GPU のピクセルパイプラインで行わせることにより処理の高速化が期待できる.

そこで，本研究では ABM と MSC による移動物体領域検出に GPU を利用することで処理を高速化し，実時間でかつ高精度な移動物体検出を行うことを提案する.

2 移動物体の検出

本稿では，屋外環境で固定カメラにより撮影されたカラー動画像から移動物体を検出することを想定する. 屋外環境において移動物体の検出を安定に行うためには，明るさの変動に対してロバストであることと，移動物体とその影を正確に分離することが必要となる. このような移動物体の検出を実現するために，色情報と空間情報それぞれに着目した 2 つの手法を組み合わせることで移動物体領域の識別を行う. 本章では，この 2 つの識別手法について詳しく説明する.

2.1 色情報を利用した識別

屋外環境において明るさの変動に対応するために，カメラから得られる画素値と太陽光のエネルギー量との関係をモデル化した明るさ可変背景モデル (Adaptive Background Model: 以下 ABM) について説明する. また，ABM を用いた識別手法について説明する.

2.1.1 明るさ可変背景モデル (ABM)[1]

屋外環境においては，太陽光以外の光源は無視することができると仮定する. このとき，屋外環境における観測光は，太陽直射光成分 (以下 直射光) と天空光成分により構成される. これらを背景成分とする. ここで，天空光成分とは太陽光が空気中の粒子等に衝突し，散乱した後に地上に届く光である. 短期的には天空光および直射光の色度の変化は小さいことから，これを無視すると，時刻 t における画素 $\boldsymbol{x} = (x, y)^\top$ において観測される対象の色 $\boldsymbol{Y}(\boldsymbol{x}, t) = (\boldsymbol{Y}_R(\boldsymbol{x}, t)\ \ \boldsymbol{Y}_G(\boldsymbol{x}, t)\ \ \boldsymbol{Y}_B(\boldsymbol{x}, t))^\top$ は，天空光成分 $\boldsymbol{Xe}(\boldsymbol{x}, t)$=$(\boldsymbol{Xe}_R(\boldsymbol{x},t)\, \boldsymbol{Xe}_G(\boldsymbol{x},t)\, \boldsymbol{Xe}_B(\boldsymbol{x},t))^\top$ と直射光成分 $\boldsymbol{Xd}(\boldsymbol{x}, t) = (\boldsymbol{Xd}_R(\boldsymbol{x}, t)\ \boldsymbol{Xd}_G(\boldsymbol{x}, t)\ \boldsymbol{Xd}_B(\boldsymbol{x}, t))^\top$ から成ると考えられる. ここで $A^\top$ は，ベクトル A の転置を表す. 複雑な反射特性を考慮せず，直射光が雲を通過するときにだけ明るさが変化すると考えれば，観測される対象の色 $\boldsymbol{Y}(\boldsymbol{x}, t)$ は式 (1) でモデル化できる.

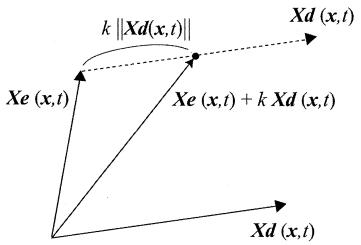


図 1: ABM の幾何学的関係

$$\boldsymbol{Y}(\boldsymbol{x}, t) = \boldsymbol{Xe}(\boldsymbol{x}, t) + k(\boldsymbol{x}, t)\boldsymbol{Xd}(\boldsymbol{x}, t) \qquad (1)$$

ここで，$k(\boldsymbol{x}, t)$ $(0 \leq k \leq 1)$ は直射光成分の混合率を表す. 式 (1) の幾何学的関係を図 1 に示す. このモデルに従えば，直射光成分の混合率 $k(\boldsymbol{x}, t)$ を明るさパラメータとして利用することで，各背景成分から成る 2 つの背景成分画像のみを用いて様々な明るさの背景画像を高精度に再現することが可能となる.

明るさが急激に変化した場合も，明るさパラメータ $k(\boldsymbol{x}, t)$ を調整するだけで様々な明るさにロバストに対応することができる. また，明るさの変動に対して各ピクセルにおいて独立に処理を行えば，移動物体の影となって直射光が遮られている部分についても対応できる.

さらに，$\boldsymbol{Xe}$ と $\boldsymbol{Xd}$ はガウス過程であると想定する.

$$\boldsymbol{Xe}(\boldsymbol{x}, t) = \boldsymbol{Xe}(\boldsymbol{x}, t-1) + \epsilon_{S_e} \qquad (2)$$

$$\boldsymbol{Xd}(\boldsymbol{x}, t) = \boldsymbol{Xd}(\boldsymbol{x}, t-1) + \epsilon_{S_d} \qquad (3)$$

ただし，$\epsilon_{S_e} \sim N(0, \Sigma_{S_e})$，$\epsilon_{S_d} \sim N(0, \Sigma_{S_d})$ は状態ノイズを表す.

2.1.2 背景と移動物体の識別

観測される対象画像の色 $\boldsymbol{Y}(\boldsymbol{x}, t)$ と前時刻の天空光成分 $\boldsymbol{Xe}(\boldsymbol{x}, t-1)$，直射光成分 $\boldsymbol{Xd}(\boldsymbol{x}, t-1)$ から構成される背景の色との相関を計算することで影除去

と移動物体検出を行う．色差ベクトルのユークリッドノルムを $D(\boldsymbol{x},t)$ として，式 (4) により算出する．

$$D(\boldsymbol{x},t)=\|\boldsymbol{Y}(\boldsymbol{x},t)-(\boldsymbol{Xe}(\boldsymbol{x},t-1)+k(\boldsymbol{x},t)\boldsymbol{Xd}(\boldsymbol{x},t-1))\| \tag{4}$$

ここで，前時刻 $t-1$ の背景成分を $\boldsymbol{Xe}(\boldsymbol{x},t-1) \approx \boldsymbol{Xe}(\boldsymbol{x},t)$，$\boldsymbol{Xd}(\boldsymbol{x},t-1) \approx \boldsymbol{Xd}(\boldsymbol{x},t)$ と仮定している．求めたユークリッドノルム $D(\boldsymbol{x},t)$ が式 (5) を満たす場合にのみ，画素 $\boldsymbol{x}$ を背景領域と識別する．

$$D(\boldsymbol{x},t) < D_{th} \tag{5}$$

ここで，D_{th} は対象画素が背景か移動物体かを決定する閾値である．ノルムが閾値 D_{th} より小さい場合は，RGB 空間上における明るさ可変背景と入力画素が非常に近接していると判断できるので，対象画素を背景と識別する．逆にノルムが閾値 D_{th} より大きい場合には非背景，つまり移動物体と識別する．

2.2 空間情報を利用した識別

ABM を用いた抽出手法では，色情報のみを用いて背景と移動物体を分離する．そのため，RGB 空間上で背景と移動物体のテクスチャの色が近接している場合は移動物体をうまく分離することができず，しばしば物体領域を背景として認識してしまう．そこで物体の検出精度の向上を図るため，MSC を用いる．

2.2.1 マージン付き符号相関 (MSC)[3]

マージン付き符号相関 (Margined Sign Correlation: 以下 MSC) は周辺増分符号相関 (Peripheral Increment Sign Correlation: 以下 PISC)[2] の拡張であり，以下の式で定義する．

$$\mathrm{MSC}_m(f(\boldsymbol{x}),g(\boldsymbol{x})) = \frac{(\mathrm{sgn}_m(f(\boldsymbol{x})),\mathrm{sgn}_m(g(\boldsymbol{x})))}{\|\mathrm{sgn}_m(f(\boldsymbol{x}))\|\cdot\|\mathrm{sgn}_m(g(\boldsymbol{x}))\|} \tag{6}$$

$$\|\mathrm{sgn}_m(f(\boldsymbol{x}))\| = \sqrt{(\mathrm{sgn}_m(f(\boldsymbol{x})),\mathrm{sgn}_m(f(\boldsymbol{x})))} \tag{7}$$

$$(\mathrm{sgn}_m(f(\boldsymbol{x})),\mathrm{sgn}_m(g(\boldsymbol{x}))) = \sum_{\boldsymbol{x}^*\in N(\boldsymbol{x})} \mathrm{sgn}_m(f(\boldsymbol{x}^*)-f(\boldsymbol{x}))\mathrm{sgn}_m(g(\boldsymbol{x}^*)-g(\boldsymbol{x})) \tag{8}$$

$$\mathrm{sgn}_m(x) = \begin{cases} +1 & |m| \le x \\ 0 & -|m| \le x < |m| \\ -1 & x < |m| \end{cases} \tag{9}$$

表 1: 背景と移動物体の識別

		chroma difference	
		background	foreground
texture difference	background	background	foreground
	foreground	foreground	foreground

ただし，(,) はベクトルの内積を表す．また，$\boldsymbol{x}^*$ は対象画素 $\boldsymbol{x}$ の近傍 $N(\boldsymbol{x})$ に含まれる点で，m はノイズを考慮したマージンを表す．$m=0$ の場合，$\mathrm{MSC}_0(f(\boldsymbol{x}),g(\boldsymbol{x}))$ は PISC と等価になる．式 (6) から分かるように，$\mathrm{MSC}_m(f(\boldsymbol{x}),g(\boldsymbol{x}))$ は $\|\mathrm{sgn}_m(f(\boldsymbol{x}))\|=0$ や $\|\mathrm{sgn}_m(g(\boldsymbol{x}))\|=0$ のときには定義できない．これは画像 f または g が近傍 $N(\boldsymbol{x})$ 内で常にノイズの影響を受けることを表し，$\mathrm{MSC}_m(f(\boldsymbol{x}),g(\boldsymbol{x}))$ の相関値は信頼性が低くなっているということを示している．

2.2.2 背景と移動物体の識別

観測される対象画像 $\boldsymbol{Y}(\boldsymbol{x},t)$ と前時刻の天空光成分 $\boldsymbol{Xe}(\boldsymbol{x},t-1)$，直射光成分 $\boldsymbol{Xd}(\boldsymbol{x},t-1)$ との輝度成分のテクスチャの違いを利用して識別を行う．輝度成分のうち G 成分がその約 6 割を占めることから，G 成分のみに着目して，式 (10) を満たす場合にのみ画素 $\boldsymbol{x}$ を背景と識別する．

$$\mathrm{MSC}_m(\boldsymbol{Y}_G(\boldsymbol{x},t)-\boldsymbol{Xe}_G(\boldsymbol{x},t-1),\boldsymbol{Xd}_G(\boldsymbol{x},t-1)) \ge E_{th} \tag{10}$$

ただし，E_{th} は背景か非背景かを区別する閾値である．ここでも $\boldsymbol{Xe}(\boldsymbol{x},t-1)\approx\boldsymbol{Xe}(\boldsymbol{x},t)$，$\boldsymbol{Xd}(\boldsymbol{x},t-1)\approx\boldsymbol{Xd}(\boldsymbol{x},t)$ の仮定を用いる．G 成分のみに着目することで輝度計算による計算コストを削減できる．

2.3 背景と移動物体の識別

色情報を利用した識別と空間情報を利用した識別の 2 つの結果に基づいて，入力画像中の画素 $\boldsymbol{x}$ が移動物体かの識別を行う．最終的な識別方法を表 1 に示す．両方とも背景と識別された場合にのみ，その画素を背景と決定する．逆にどちらか一方でも移動物体と識別された場合には，背景と識別したもう一方の識別結果を無視して，その画素を移動物体と決定する．

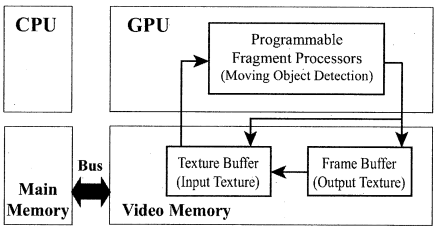


図 2: GPU でのデータの流れ

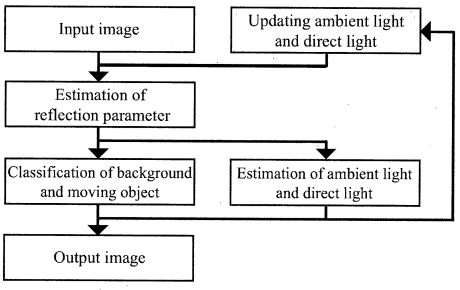


図 3: システムの流れ

3 GPU を用いた移動物体検出システムの実現

本章では，提案する GPU を用いた移動物体検出システムについて説明する．まずシステムの概略を説明し，続いて GPU での処理の詳細について示す．

3.1 GPU を用いた移動物体検出システム

GPU は，本来，3D グラフィックス処理専用の LSI であったが，近年の急速な性能向上とプログラマブル化により，GPU を用いることで汎用的な計算を高速化する試みが注目されている [4, 5]．GPU 上のプログラマブルパイプラインは，3 次元空間における幾何学的計算を処理する頂点プロセッサと，2 次元画像を処理するピクセルプロセッサから構成されており，GPGPU では主にフラグメントプロセッサを使用する．GPU のアーキテクチャと GPGPU のプログラムモデルの概念を図 2 に示す．

本システムでは，GPU において，メインメモリから読み込まれたフレームに対して各ピクセルごとに背景か移動物体かの識別を行い，識別結果のテクスチャをメインメモリに出力する．GPU 機能を利用するための開発ツールに Quartz Composer を使用し，GPU での処理を記述する言語に Objective-C を用いる．また，テクスチャに各ピクセル値が単精度浮動小数点数の float テクスチャを用いた．

3.2 GPU での処理

本システムの GPU での処理の概要を図 3 に示す．システムは移動物体の検出処理と背景成分の更新処理により構成される．

まずはじめに，時刻 t における画素 $\boldsymbol{x}$ での明るさパラメータ $k(\boldsymbol{x},t)$ を，時刻 t の入力画像 $\boldsymbol{Y}(\boldsymbol{x},t)$ と前時刻 $t-1$ の背景成分 $\boldsymbol{Xe}(\boldsymbol{x},t-1)$ と $\boldsymbol{Xd}(\boldsymbol{x},t-1)$ を用いて次式により算出する．

$$k(\boldsymbol{x},t) = \frac{((\boldsymbol{Y}(\boldsymbol{x},t) - \boldsymbol{Xe}(\boldsymbol{x},t-1)), \boldsymbol{Xd}(\boldsymbol{x},t-1))}{\|\boldsymbol{Xd}(\boldsymbol{x},t-1)\|^2} \tag{11}$$

ここでも $\boldsymbol{Xe}(\boldsymbol{x},t-1) \approx \boldsymbol{Xe}(\boldsymbol{x},t)$，$\boldsymbol{Xd}(\boldsymbol{x},t-1) \approx \boldsymbol{Xd}(\boldsymbol{x},t)$ の仮定を用いる．明るさパラメータが求まると，各ピクセルについてそれぞれ背景か移動物体かの識別を行う．この識別には色情報及び空間情報を用いる．

一方で，明るさパラメータ $k(\boldsymbol{x},t)$ から背景成分の観測値 $\boldsymbol{Ye}(\boldsymbol{x},t)$ と $\boldsymbol{Yd}(\boldsymbol{x},t)$ をそれぞれ式 (12)(13) により推定する．

$$\boldsymbol{Ye}(\boldsymbol{x},t) = \boldsymbol{Y}(\boldsymbol{x},t) - k(\boldsymbol{x},t)\boldsymbol{Xd}(\boldsymbol{x},t-1) \tag{12}$$

$$\boldsymbol{Yd}(\boldsymbol{x},t) = \frac{\boldsymbol{Y}(\boldsymbol{x},t) - \boldsymbol{Xe}(\boldsymbol{x},t-1)}{k(\boldsymbol{x},t)} \tag{13}$$

得られた観測値を用いてカルマンフィルタに適用することで，時刻 t における背景成分の状態 $\boldsymbol{Xe}(\boldsymbol{x},t)$ と $\boldsymbol{Xd}(\boldsymbol{x},t)$ を推定する．このときの観測方程式をそれぞれ式 (14)(15) に示す．

$$\boldsymbol{Xe}(\boldsymbol{x},t) = \boldsymbol{Ye}(\boldsymbol{x},t) + \varepsilon_{O_e} \tag{14}$$

$$\boldsymbol{Xd}(\boldsymbol{x},t) = \boldsymbol{Yd}(\boldsymbol{x},t) + \varepsilon_{O_d} \tag{15}$$

ただし，$\varepsilon_{O_e} \sim N(0,\Sigma_{O_e})$，$\varepsilon_{O_d} \sim N(0,\Sigma_{O_d})$ はそれぞれ観測ノイズを表す．このとき，背景と移動物体の識別において背景と識別された画素についてのみ，背景成分を更新する．また，直射光成分 $\boldsymbol{Xd}(\boldsymbol{x},t)$ は，影領域などの直射光成分がほとんど含まれていない領域を避け，以下の条件式を満たす場合にのみ更新を行う．

$$k(\boldsymbol{x},t) \geq K_{th} \tag{16}$$

以上の全処理を各画素 $\boldsymbol{x}$ について行う．

システム動作開始時には，背景成分 $\boldsymbol{Xe}(\boldsymbol{x}, t-1)$ と $\boldsymbol{Xd}(\boldsymbol{x}, t-1)$ に適当な初期値を設定する．システム動作開始以降に背景成分の初期値推定期間を設け，以降を移動物体の検出期間とする．

4 実験

提案システムを用いて実際に実験を行い，精度と計算速度を評価した．MSC を適用する画素 $\boldsymbol{x}$ の近傍領域のサイズは，精度と計算速度の両方に影響することが考えられる．そこで，近傍領域の窓サイズを変化させて精度と処理時間を測定し，適当な窓サイズを検討した．また，空間情報を利用した識別で，輝度成分を G 成分で代用していることによる精度と計算速度への影響について検討した．

実験環境を以下に示す．

- OS : Mac OS X version 10.5.4
- CPU : Quad-Core Intel Xeon (2 x 2.8GHz)
- RAM : 4GB
- GPU : GeForce 8800 GT
- VRAM : 512MB

実験には解像度 720×486 の 8bit カラー動画像を用いた．処理時間の測定には，GPU を利用したシステムは動画像ファイルを入力として fps 値を測定し，CPU のみを用いたシステムは同じ動画像内の 1 フレームを入力としてその処理時間を測定した．なお，GPU の処理には動画像のデコード時間も含まれている．閾値 D_{th} と E_{th} はそれぞれ 0.06，0.5 と設定した．また，MSC のマージン m を 0.008 と設定した．これらの値は経験的に設定した．

4.1 実験結果

図 6，7 に実際に CPU のみを用いたシステムで処理を行ったときの入力画像を示す．CPU のみを用いたシステムでは，移動物体の有無により処理時間が変化することから，移動物体を多く含むフレームと移動物体を含まないフレームの 2 つを入力画像として選択した． 表 2 は MSC の窓の一辺を 1, 3, 5, 7, 9pixel の 5 通りに変化させたときの 1 フレームあたりの処理時間の実測値を，図 4 はそれをグラフにプロットしたものを示す．表 3 は空間情報を利用した識別で輝度成分，G 成分を用いた場合の 1 フレームあたりの処理時間の実測値を，図 5 はそれをグラフにプロットしたものを示す．また，図 8,9 に窓の一辺

表 2: システム全体の処理時間の比較

window size	only ABM	3 x 3	5 x 5	7 x 7	9 x 9
CPU (objects)	1.50	1.63	2.47	3.24	3.12
CPU (background)	1.84	1.96	2.30	2.78	3.41
GPU	0.08	0.09	0.09	0.11	0.33
CPU / GPU	20.8	21.1	23.7	23.6	9.8

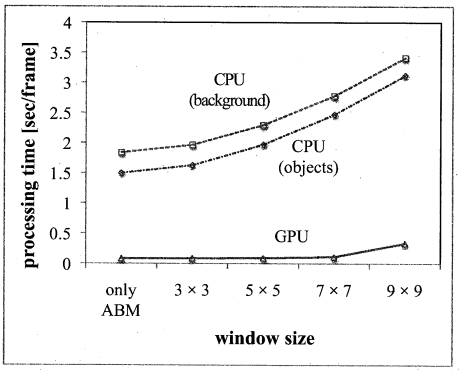


図 4: MSC の計算手法による処理時間の比較

表 3: MSC の計算手法による処理時間の比較

window size	3 x 3	5 x 5	7 x 7	9 x 9
Luminance	0.09	0.10	0.13	0.44
G	0.09	0.09	0.11	0.33

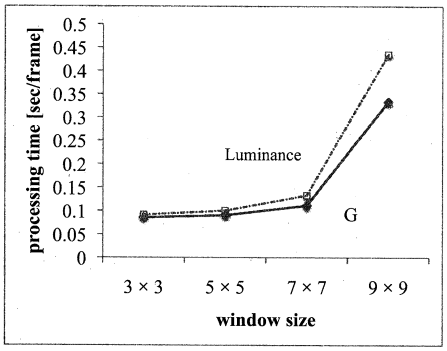


図 5: MSC の計算手法による処理時間の比較

表 4: MSC の計算手法による検出精度の比較

		object region		background region		removal region	
		pixels	rate (%)	pixels	rate (%)	pixels	rate(%)
3 x 3	G	47285	**72.13**	265849	**96.63**	9009	**97.39**
	Luminance	51778	**78.99**	264273	**96.06**	8776	**94.88**
5 x 5	G	48157	**73.46**	267210	**97.13**	9073	**98.09**
	Luminance	48069	**73.33**	267797	**97.34**	9084	**98.21**
7 x 7	G	49460	**75.45**	267502	**97.23**	8962	**96.89**
	Luminance	52024	**79.36**	264438	**96.12**	8709	**94.13**

を 3, 5, 7pixel の 3 通りに変化させた場合の処理結果を示し，表 4 にそれぞれの移動物体領域と背景領域の検出率，及び影の除去率を示す．これらの図において，白で表示されている部分は背景と識別された画素である．図 6 において，移動物体領域の画素数は 65554pixel，背景領域の画素数は 274749pixel，影領域の画素数は 9250pixel である．このときの検出精度と処理速度の関係を図 11 に示す．図 10 は，窓サイズを 7×7 とした場合の様々なシーンにおける提案システムによる処理結果を示す．

4.2 考察

4.1 で得られた実験結果を用いて考察を行う．GPU を利用した提案システムと CPU を用いたシステムとの処理速度の比較を行い，MSC の近傍領域の窓サイズと G 成分の代用による精度，計算速度への影響について検討し，今後の課題について述べる．

4.2.1 CPU との比較

表 2 および図 4 より，GPU を用いることで処理が高速化されていることが分かる．MSC の窓サイズにより計算時間の比率は異なるが，計算時間の比率が最小の場合にも GPU は CPU より 9 倍以上速く計算できている．一方，GPU のみに着目すると，MSC の窓サイズが大きくなるにつれ計算時間が 2 次関数状に増加している．これは，窓の一辺が大きくなると MSC の計算に用いる画素数がその 2 乗倍されるためである．しかし，窓の一辺が 9pixel の場合は検出結果が 3fps で出力されるため不自然に見えるが，7pixel 以下であれば十分に連続的な検出結果動画像が得られる．

4.2.2 MSC の計算方法による影響

表 3 および図 5 より，MSC の計算に G 成分のみを用いることで処理速度の向上が見られる．窓サイズが 5×5 以下の場合では両手法に差はほとんど見られないが，7×7 以上になるとその差が明らかになる．これは，輝度成分を用いる場合には窓内の画素数分の輝度計算が伴い，窓サイズが大きくなるにつれその回数が増加するためである．

次に表 4 および図 8, 9, 11 から，精度について検討する．まず，MSC の計算方法による精度への影響を考える．表 4 より，移動物体領域の検出率は輝度成分を用いた方が高い一方で，影の除去率は G 成分のみを用いた方が高い．また，検出率，除去率の差は共に微小である．以上のことから，MSC の計算に G 成分値のみを用いても精度には支障がないことが言える．次に，G 成分のみを用いた場合の窓サイズによる精度への影響について考える．図 11 より，窓サイズが大きくなるにつれ，移動物体領域の検出率が向上する一方で影領域の除去率は低下していることが見てとれる．また，図 8 を見ると，画像右上部の女性の膝部分や左下部の女性の頭部などにおいて検出精度が向上しており，影領域の除去率の低下は画面右下部の女性の足下などに見られることが分かる．しかし，影領域の除去率の低下による影響は，移動物体領域の検出率の向上と比較して微小であり無視しても差し支えない．以上のことから，今回用いたシーンにおいて最適な窓サイズは 7×7 であると言える．

4.2.3 今後の課題

今回の実験では，天候に大きな変動がない 7 秒程度の短い動画像を用いた．しかし，実環境への適応を考えると，長時間の検出で精度の維持ができることや天候の変動，例えば雨の中や日出時での移動物体検出に対応できることが必要となる．本論文で提案したシステムでは色情報，空間情報を用いた識別での閾値 D_{th}，E_{th} や MSC のマージン m は一定であると仮定しているが，天候の変動により明るさが大きく変化すれば，それに伴いこれらのパラメータの値が変動する必要があると考えられる．また，長時間の移動物体検出を行う際に，背景成分が途中で発散することなく安定に推定され続けなければならない．今回の実験においても，カルマンフィルタのパラメータの設定によっては背景成分の推定値が検出処理中に発散してしまうことがあった．

本論文で提案したシステムは，GPU を用いて移動物体の検出処理を実時間化することでオンライン処理が可能となり，長時間の移動物体検出に対応できる．そこで，今後は提案システムを用いて様々な条件下でより長時間の実験を行い，環境の変動にロバストなアルゴリズムを検討する必要がある．

5 おわりに

本稿では，GPU を用いることで移動物体領域の検出処理を高速化し，リアルタイムで高精度な移動物

図 6: 入力画像 (移動物体あり)

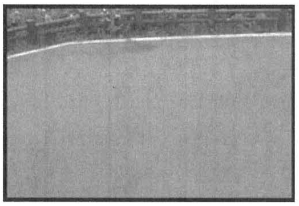
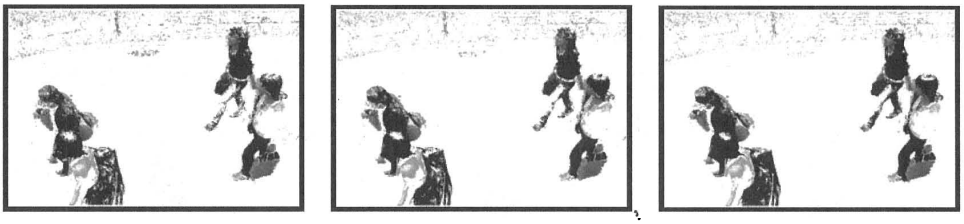
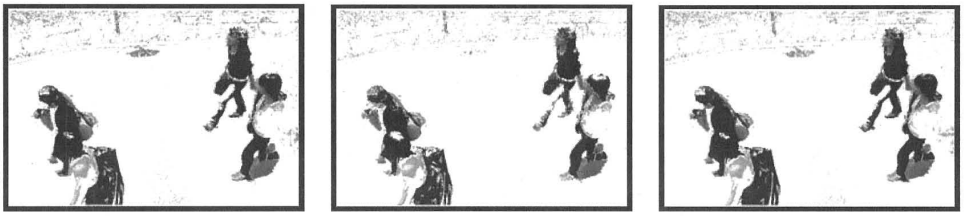


図 7: 入力画像 (移動物体なし)



(a)3×3　　(b)5×5　　(c)7×7

図 8: 空間情報を利用した識別に G 成分を用いた場合の処理結果



(a)3×3　　(b)5×5　　(c)7×7

図 9: 空間情報を利用した識別に輝度成分を用いた場合の処理結果

(a)　　(b)　　(c)

図 10: 窓サイズを 7×7 とした場合の処理結果

体検出を行うことを提案した．また，MSC の計算に G 成分のみを用いることで更なる処理速度の向上を実現した．そして，実際に屋外環境で撮影された動画像を用いて実験を行い，提案システムと CPU を用いたシステムの処理速度とを比較することで，提案システムの有効性を確認した．

今後は，実環境を想定した様々な状況下で高精度な検出が行えるよう，アルゴリズムを検討していく必要がある．

参考文献

[1] 伊藤竜之介, 岩井儀雄, 谷内田正彦, “照明変動と影の影響を抑えた移動物体の検出”, 情報学研報. No.127-17 pp.119-126, 2001.

[2] S. Kaneko, Y. Satoh, S. Igarashi, “Roust object detection image sequence using peripheral increment sign correlation”, Proc. of the 5th Japan-France Congress on Mechatronics, pp.287-292, 2001.

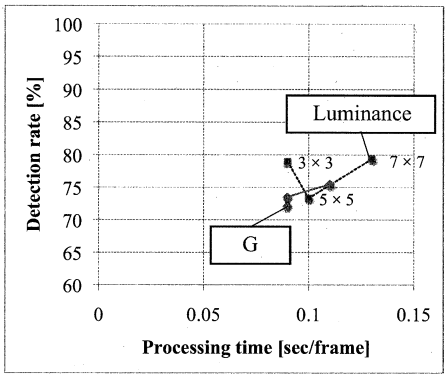
[3] 吉村浩典, 岩井儀雄, 谷内田正彦, “屋外環境における明るさ可変背景と MSC を用いた移動物体検出”, 電子情報通信学会論文誌. D Vol. J90-D No.8 pp.1987-1997, 2007.

[4] J. D. Owens, et al, “A survey of generalpurpose computation on graphics hardware”, Eurographics, State of the Art Reports, pp. 21- 51, 2005.

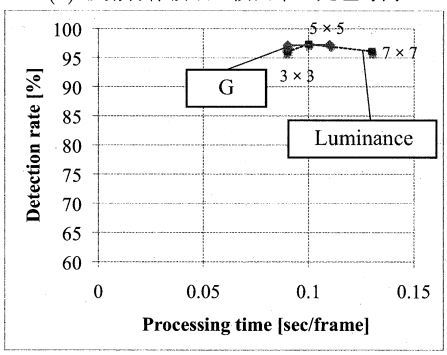
[5] L. Zhang, R. Nevatia, “Efficient scan-window based object detection using GPGPU”, 2008. CVPR Workshops 2008. IEEE Computer Society Conference on, pp. 21- 51, 2005.

[6] A. Price, J. Pyke, D. Ashiri, T. Cornall, “Real Time Object Detection for an Unmanned Aerial Vehicle using an FPGA based Vision System”, Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on , vol. 84, pp. 1931-1940, 2004.

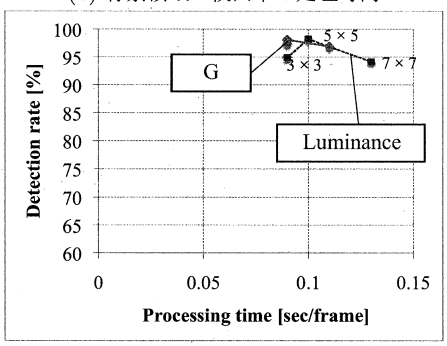
[7] L. L. Presti, M. L. Cascia, “Real-Time Object Detection in Embedded Video Surveillance Systems”, 2008. WIAMIS ’08. Ninth International Workshop on, pp. 151-154, 2008.



(a) 移動物体領域の検出率と処理時間



(b) 背景領域の検出率と処理時間



(c) 影の除去率と処理時間

図 11: MSC の計算手法による検出精度と処理時間の関係