移動カメラ画像データベースからの消失点軌跡推定処理の MapReduce による並列化

山本　宗†　　金子　邦彦‡

†九州大学大学院システム情報科学府　〒819-0395　福岡市西区元岡 744 番地
‡九州大学大学院システム情報科学研究院 〒819-0395　福岡市西区元岡 744 番地
E-mail:　†yamamoto@db.is.kyushu-u.ac.jp,　‡kaneko@ait.kyushu-u.ac.jp

あらまし　近年，カメラの軽量化・小型化，低価格化により，車載カメラが車両に搭載される傾向にある．そこで，車載カメラから撮影された時系列画像を解析して，外界の事物までの距離や特性を解析することは重要な研究課題である．また, 車載カメラにより蓄積された大量の動画像を処理する場合に, 単体のコンピュータでは処理性能の向上に限界があり，多くの処理時間を費やしてしまう．本論文では，車両の運動分析を行うために，時系列画像の特定の領域に対して，画像ピラミッドを利用した Lucus-Kanade アルゴリズムを用いてオプティカルフローを算出し，さらに，動きの消失点を推定する．また，撮影された大量の動画像処理時間を短縮するために分散並列処理が可能なプログラミングモデルである MapReduce を用いて処理の並列化の目指す.

キーワード　データベース，移動カメラ，動画像処理，動きの消失点，MapReduce

Parallel Estimation Processing of the Vanishing Point Trajectories from Image Database by Moving Camera with MapReduce

Muneto Yamamoto†,　Kunihiko Kaneko‡

† ‡ Dept. of Information Science and Electrical Engineering, Kyushu University of Graduate School
Motooka 744, Nishi-ku, Fukuoka-shi, Fukuoka 819-03953 Japan
E-mail:　† ‡ {yamamoto, kaneko}@ait.kyushu-u.ac.jp

1. はじめに

近年では，車載カメラより得られた動画像処理をおこない，動画像から視覚情報を解析する研究も行われている．動画像とは，図 1 に示すように，時刻を示す数値あるいは全順序が付いた何らかの属性値を属性として持つフレームの並びである．ここで，車載カメラから得られる時系列画像を用いて，自動車の挙動，障害物の判定，交通渋滞・道路状況を，画像処理を行うことで，事故の防止に繋がる様々な研究が行われている[7],[12],[13]．また，車載カメラから撮影される時系列画像は観測系である車両の運動による影響を受ける．車両の運動を GPS やジャイロ，または加速度センサにより車両の情報を付加して画像処理を行う手法も存在する[2]．しかし本論文では，観測系である車両の運動は未知である場合で行うものとする．また，PC のメモリ価格の低下，大容量化に伴い，車載カメラにより得られる動画像が大量に蓄積され，得られた動画像を動画像データベースシステムに保持し，検索，分析等の処理を行うとき，単体のコンピュータでは処理性能の向上に限界がある．従来から，大規模・動画像データベースの並列分散処理方式について，種々の研究が行われており，近年では，プログラミングの容易さ等から，大量のデータを複数のマシンに分散して処理できるオープンソースのクラウド基盤システムである Hadoop 上で, MapReduce の機能を用いて種々のデータベース処理を行う試みも研究されている．

本論文では，ある地域の都市景観の動画像を車載カメラより撮影して作られた動画像データベースについて，車両の運動分析を行うために，車載カメラより得られた時系列画像に関して処理を行う．そして，特定の領域に対してオプティカルフローを算出し，動きの消失点を推定し結果を確認する．また，動画像処理を行う際に問題となる動画像処理時間の短縮をするために分散並列処理が可能なプログラミングモデルである MapReduce において Ruby 言語を用いて時系列画像処理の並列化の実現を目指す．

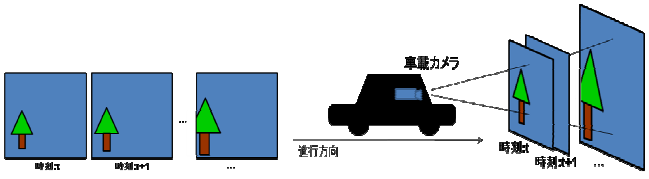


図 1　動画像の各フレームと車載カメラ

本論文の構成は以下のようになっている．まず，第 2 章では提案手法に関連する画像処理について説明する.第 3 章では提案手法に対して MapReduce のオープンソース実装のクラウド基盤システム Hadoop においての並列化について述べる．そして，第 4 章では実験の結果を示し，第 5 章においてまとめと今後の課題について述べる．

2. 提案手法

車載カメラから得られた時系列画像について，撮影時の照明条件の変動，濃度勾配が平坦な領域の混在，街路樹など不規則な模様のある領域の混在などの理由により動き解析を画像全域に渡って正確なオプティカルフローを算出することは難しい．そのため，時系列画像に対する照明条件の緩和，そして，精度よく原画像のオプティカルフローを算出するために，原画像の画像ピラミッドを生成する．また，一定領域を切り出し，その領域に対して，さらに，特徴量の少ない領域を除去した範囲でオプティカルフローの算出を行う．しかし，精度を上げる為にガウシアンフィルタを適用した後に，ダウンワンプリングにより作成した画像ピラミッドを利用した Lucas-Kanade 法を用いる[8]．また，求めたオプティカルフローを基に動きの消失点を推定する．

2.1. 画像処理

まず，照明条件の緩和を行うために，ヒストグラム平坦化を行う．ヒストグラム平坦化により輝度分布の範囲を広げることができ，分布の偏りの緩和をおこなえる．つまり，時系列画像全体の照明条件の変動を緩和させ，動き解析の精度を改善する．ここで，画像の各画素における階調値を a，各階調数における出現数を $P(a)$で表す．a の範囲は 0 から 255 である．そして，画像の幅と高さをそれぞれ M,N とし，総画素数 Z とする．修正後の階調値を b として平坦化のための式(2.1)を以下に示す．

$$b=\frac{\sum_{x=0}^{a}P(x)}{Z}\times 255 \quad (2.1)$$

ここで，総画素数 $Z=M\times N$ である．式(2.1)より画像の各画素に対して階調値 b を求める．車載カメラにより撮影された画像に対してヒストグラム平坦化を行った例を図 2 に示す．

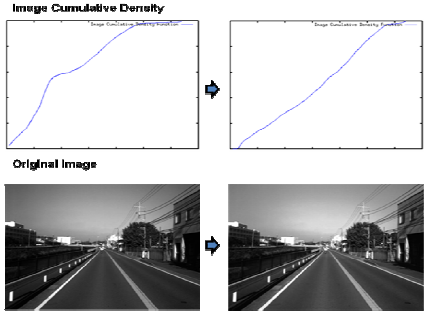


図 2　輝度の出現累積分布と
画像のヒストグラムの平坦化の例

次に，画像の一定領域を切り出した範囲に対して，特徴量の抽出を行う．ここでの画像の特徴量とは画像の固有値の大小関係より定まるコーナーやエッジであればあるほど，その画素は特徴量があると定める．また，特徴量のない画像の濃淡値は平坦である．特徴量抽出画像は 256 階調のグレースケール画像とし，特徴量の最も多い画素の濃淡値を 0，つまり黒で表し，特徴量の最も低い画素の濃淡値を 255，つまり白として原画像の特徴量を出力する．ここで，まず画像のコーナーやエッジを求めるために各画素の周辺画素である小領域の相違度の 2 乗和を $S(i,j)$と定めて，注目した画素を(i_0,j_0)として，そこを中心としたウィンドウサイズを W とする．$S(i,j)$はグレースケール画像 G の各画素 $g(i,j)$を用いて以下の式(2.2)で表せる．

$$S(i,j)=\sum_{u,v\in W}\left(g(i_0+di+u,j_0+dj+v)-g(i_0+u,j_0+v)\right)^2\times u(u,v) \quad (2.2)$$

グレースケール画像の各画素 $g(i,j)$の i 方向，j 方向の微分 g_i,g_j を用いて以下の式(2.3)で表せる．

$$S(i,j)=\sum_{u,v\in W}\left(g_i(i_0+u,j_0+v)di+g_j(i_0+u,j_0+v)dj\right)^2\times u(u,v) \quad (2.3)$$

式(2.3)を展開すると

$$S(i,j)=\sum_{u,v\in W}\left(u(u,v)\left\{g_i^2di^2+2g_i(i_0+u,j_0+v)\times g_j(i_0+u,j_0+v)didj+g_j^2dj^2\right\}\right)$$

$$=[di,dj\begin{bmatrix}\sum u(u,v)g_i^2 & \sum u(u,v)g_ig_j\\ \sum u(u,v)g_ig_j & \sum u(u,v)g_j^2\end{bmatrix}\begin{bmatrix}di\\dj\end{bmatrix} \quad (2.4)$$

と展開される．ここで，画像に対して特徴量の抽出を行うために，はじめに注目した画素の周辺画素の階調値を微分もしくは差分を用いた演算により画像の線や縁といった輪郭を強調する 1 次微分フィルタである Prewitt フィルタを用いて自己相関行列を作成する．画像の x 方向と y 方向，つまり画像の横方向である i 方向，画像の縦方向である j 方向に対して 1 次微分をする．また，i 方向微分の 2 乗，j 方向微分の 2 乗，i 方向微分と j 方向微分とを掛け合わせたものに対して，それぞれガウシアンフィルタによる平滑化をおこない，結果をそれぞれ A,B,C とすると以下の式(2.5)で表せる．

$$A=(I_i)^2=\left(\frac{\partial I_k}{\partial i}\right)^2,\ B=(I_j)^2=\left(\frac{\partial I_k}{\partial j}\right)^2,\ C=I_iI_j=\left(\frac{\partial^2 I_k}{\partial i\partial j}\right) \qquad (2.5)$$

各画素値に対して自己相関行列 M を A,B,C を用いて以下のように定める．

$$M=\begin{bmatrix} A & C \\ C & B \end{bmatrix} \qquad (2.6)$$

ここで，Harris Corner Detection の特徴点抽出の式として以下の式(2.7)を示す[4],[9].

$$c(i,j)=\det(M)-k(tr(M))^2 \qquad (2.7)$$

det は行列式を表し，tr は対角成分の和を表す．だだし，k は調節できるパラメータとし，一般的に k の値は 0.04~0.06 の範囲である．今回は k の値を 0.04 とする．また，固有値 λ_1,λ_2 は下式より求められる．

$$\det(M)=\lambda_1\lambda_2=AB-C^2 \qquad (2.8)$$

$$tr(M)=\lambda_1+\lambda_2=A+B \qquad (2.9)$$

$c(i,j)$の値に対して閾値 T を設定して，$c(i,j)<T$ となる場合に $c(i,j)=0$ としてコーナー検出を行う．また，$c(i,j)$と特徴点との関係については，$c(i,j)>0$ の場合はコーナー，$c(i,j)<0$ の場合はエッジ，$c(i,j)\approx 0$ の場合は平面部分となり $c(i,j)$が 0 に近づくにつれて特徴量がない点とみなすことができる．固有値 λ_1,λ_2 より計算される各画素に対する $c(i,j)$の絶対値をとり，その値を $R(i,j)$ とおき，最小値を $min(R(i,j))$，最大値を $max(R(i,j))$と書き表して最小値と最大値の範囲で画像 F の各画素に対する $R(i,j)$を 0 から 255 までの範囲で 256 階調に割り当てる．各画素の中での最大値をとる点の画素を 255，最小値をとる点の画素を 0 として画像処理を行うことで画像のコーナー，エッジといった特徴点を抽出した画像を作成する．特徴量抽出画像では，濃淡値の濃い部分が特徴量の強い画像である．

2.2. 画像ピラミッド

画像ピラミッドとは，低い解像度の画像から段々高くなる解像度の画像のセットであり，解像度を上げながら計算を繰り返すことにより，効率的に正確な処理結果を得ることができる．そして，画像ピラミッドの例を図 3 に示す．画像ピラミッドの各画像を使い分けることで，各々の判別に最適な解像度の画像から処理結果を算出することができる．

また，局所的な特徴量を高解像度画像から抽出した場合，ユーザが認識し得ないほど細かな部位の情報しか含まない可能性がある．逆に，低解像度すぎると，利用者が主観的に類似度を判定する際に重視する部位の情報が欠落してしまう可能性もある．つまり，最適解像度は一意に定まるものではなく対象によって変化する．

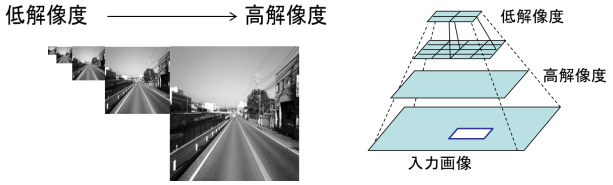


図 3　画像ピラミッド例

オプティカルフローの推定法の 1 つである勾配法

本研究では，4 種類の解像度(1/1, 1/2, 1/4, 1/16) の画像で構成される画像ピラミッドからオプティカルフローを算出する．

2.3. オプティカルフロー

オプティカルフローとは，動画像の時間的，そして空間的な輝度分布の変化を使い，画素ごとに動きベクトルを求めてできた結果である．例えば図 4 のように，注目画素(i,j)が別のフレームにおいて別の場所に動いたとき，その移動を表すベクトルが，この注目画素の動きベクトルである．

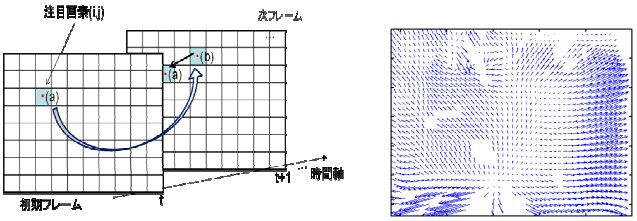


図 4　オプティカルフローの概形と表示例

オプティカルフローの推定法の 1 つである勾配法は，画像の局所的な特徴である時間勾配，および空間勾配を関係付ける式をもとに動きベクトルの算出を行うものである．勾配法には被写体が動いた時に任意の点の濃淡値は変化をしないという条件より求まる基本拘束式があるが，オプティカルフローを算出するには，各画素のフロー成分を縦方向と横方向の 2 つの未知数を求めなければならず，第 2 の拘束条件として今回使用した手法としては勾配法の中の空間的局所最適法である[10],[11]．空間的局所最適法は，同一物体の濃淡値が任意の空間領域においてオプティカルフローをほぼ一定に保つという条件で解く方法である．基本拘束式と第 2 の拘束条件を説明する．

基本拘束式を求めるために，ある時刻 t において画像の各画素の位置を(x,y)とし，画像の任意の点である位置(x,y)における濃淡値を $I(x,y)$とする．被写体が動いた時に，任意の点の濃淡値は変化をしないと仮定すると，次式(2.10)が得られる．

$$I(x,y,t)=I(x+\Delta x,y+\Delta y,t+\Delta t) \qquad (2.10)$$

ここで，Taylor 級数展開をおこない，2 次以上の高次項は微小であるとして無視することにより式(2.11)で表せる．

$$I(x,y,t)=I(x,y,t)+\frac{\partial I}{\partial x}\Delta x+\frac{\partial I}{\partial y}\Delta y+\frac{\partial I}{\partial t}\Delta t \qquad (2.11)$$

ここで，オプティカルフローの成分 x,y をそれぞれ動きベクトル u,v で表す(2.12).

$$u=\frac{dx}{dt},v=\frac{dy}{dt} \qquad (2.12)$$

従って，以下の基本拘束式(2.13)が導かれる[10].

$$I_x u+I_y v+I_t=0 \qquad (2.13)$$

基本拘束式と速度空間の関係と開口問題を図 5 に示す. 式(2.13)より得られる動画像中の各画素において求まるオプティカルフローの x 方向と y 方向の微分である 2 つの成分 u,v は未知数であるために, 基本拘束式 1 つでは u,v を一意に定めることができない．このことを開口問題という.

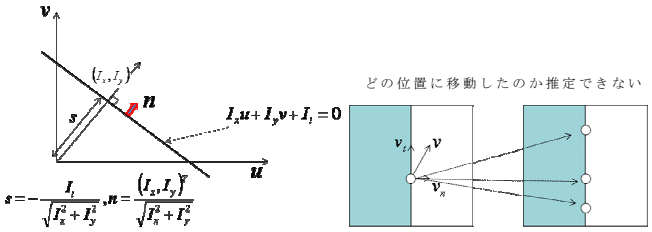


図 5　速度空間と速度空間

開口問題とは，輝度勾配が平坦である場合，あるいは一方向にしか存在しない場合には，対応する画素が複数存在するためにオプティカルフローを推定できないという問題である.

また，基本拘束式 1 つでは開口問題などが生じるために，u,v を一意に定めることができない．ここで，基本拘束式を解くために空間的局所最適法において，局所領域内でのオプティカルフローが一定という拘束条件を定めると，第 k 番目の画素について，次式(2.14)が成り立つ[10],[13].

$$I_{xk}u+I_{yk}v=-I_{tk} \qquad (2.14)$$

空間領域として $n\times n$ pixel の正方形を考慮すると

$$Ap=-b$$

ここで，A,p,b はそれぞれ以下の式(2.15)である.

$$A=\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn^2} & I_{yn^2} \end{bmatrix},p=\begin{bmatrix} u \\ v \end{bmatrix},b=\begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \\ I_{tn^2} \end{bmatrix} \qquad (2.15)$$

また，式変形より式(2.16)が成り立つ.

$$\hat{p}=-(A^tA)^{-1}A^tb \qquad (2.16)$$

よって，拘束条件を付加することでオプティカルフローの成分を推定できる．ここで，使用した空間領域の範囲 $n\times n$ pixel は 15×15 として求めている.

2.4. 動きベクトルの消失点

車載カメラから得られる動画像において，オプティカルフローを延長した直線による交点を用いて，消失点を求めることができる．この消失点は画像上の消失点ではなく，動きの消失点のことであり，画像上の消失点とは違うものである．カメラが一定の方向に向かって進んでいる場合のカメラワークと消失点を図 6 に示す．動きの焼失点の軌跡により，車載カメラ自体の動きを推定することができる[6]．また，消失点と理想的な動きベクトルとの関係を図 7 に示す.

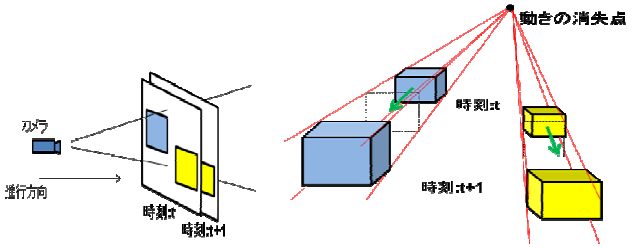


図 6　カメラワークと動きの消失点

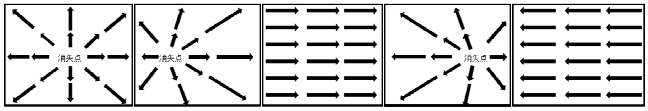


図 7　直進，左カーブ，左折，右カーブ，右折の理想的な動きベクトル

ここで，消失点を求めるために画像の各画素の位置 (x,y) において, オプティカルフロー $p=[u,v]^t$ が求められているとする．各画素から各動きベクトルの直線の交わる点，つまり消失点を求めるために直線の方程式を以下の式(2.17)で表す.

$$ax+by=d \qquad (2.17)$$

ここで，a,b,d はそれぞれ以下の式(2.18)である.

$$a=-\frac{v}{\sqrt{u^2+v^2}},b=\frac{u}{\sqrt{u^2+v^2}},d=\frac{uy-vx}{\sqrt{u^2+v^2}} \qquad (2.18)$$

また，各画素の動きベクトルを考慮すると

$$Cq=d$$

ここで，C,q,d はそれぞれ以下の式(2.19)である.

$$C=\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_{length} & b_{length} \end{bmatrix},q=\begin{bmatrix} x \\ y \end{bmatrix},d=\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{length} \end{bmatrix} \qquad (2.19)$$

ここで，$length$ はオプティカルフロー対応する各画素の縦×横の範囲で，動きベクトルの数と一致する．式変形より式(2.20)が成り立つ.

$$\hat{q}=-(C^tC)^{-1}C^td \qquad (2.20)$$

よって，$q=[u,v]^t$ を求めることができる．動きの分析を行うために，理想の動きベクトルの場合の縦軸に動きベクトルの極座標系の 0 から 360 度をとり，横軸に今回使用した画像の x 座標 0 から 640 と y 座標 0 から 480 までをとった場合のグラフを示す．また，位置ベクトルを $(x,y)=(vector\_x,vector\_y)$ と表すとベクトルの大きさと回転角はそれぞれ以下の式(2.21),(2.22)で表せる．回転角を θ とおくときの動きベクトルの極座標系と，回転角と x, y との関係をグラフ化すると図 8 のようになる．

$$\sqrt{vector\_x^2 + vector\_y^2} \qquad (2.21)$$

$$\theta = \arctan\left(\frac{vector\_y}{vector\_x}\right) \qquad (2.22)$$

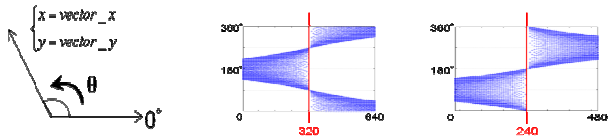


図 8　回転角の座標系，x 方向または y 方向と回転角の関係

ここで，オプティカルフロー抽出例として，車載カメラを設置した自動車が並進運動をおこなっている場合とカーブ時の場合での動きベクトルをそれぞれ図 9 と図 10 に示す．

(a)並進運動

図 9　連続 3 フレーム間の動画像の並進運動時の動きベクトル抽出画像

(b)カーブ

図 10　連続 3 フレーム間の動画像のカーブ時の動きベクトル抽出画像

3. Hadoop

MapReduce と HDFS(Hadoop Distributed File System)の 2 つのコンポーネントから構成されており，単一のマスタサーバと複数のスレーブサーバにより分散処理を行っており，MapReduce の要素といて，JobTracker や TaskTracker があり，HDFS の要素として NameNode や DataNode が含まれる．図 11 に構成の概形を示す．

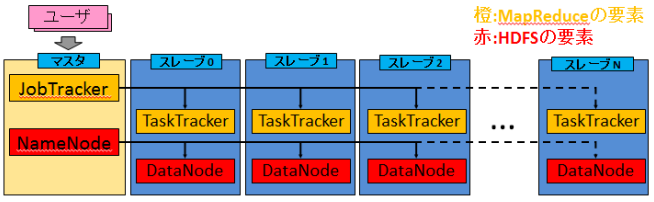


図 11　MapReduce と HDFS の要素による構成

3.2. MapReduce

MapReduce とは大規模データを分散並列処理するためのフレームワークであり，Google のオープンソースのクラウド基盤システムである Hadoop 上で実装されている[3],[9]. プログラミングにおいて，Map, Shuffle, Reduce と呼ばれる各ステップを記述することで分散並列処理が可能となる．図 12 に Map と Reduce 実行処理の流れを示す．また，分散・並列プログラミングにおいて問題となる，ロードバランス，プロセス間通信，ディスク容量，耐障害性といった点に関して自動で処理を行うため，ユーザは Map 関数と Reduce 関数のアルゴリズムの記述により分散並列処理を行える．しかし，効率よく処理するには並列計算，データの分散配置，key-value 対の割り当てなど複雑な問題が多数存在する．

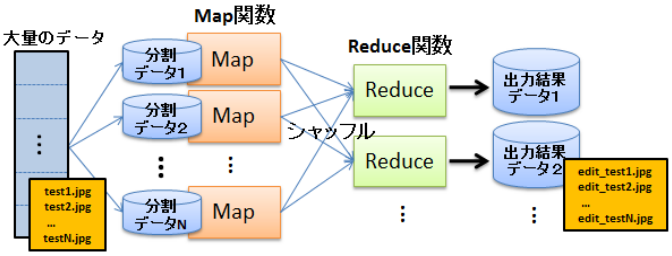


図 12　画像処理時の Map と Reduce 実行処理の流れ

1. Map 処理

入力データ（キーと値のペア）を受け取り，任意の形式に変換し必要な情報を抽出し，並列実行可能．

2. Reduce 処理

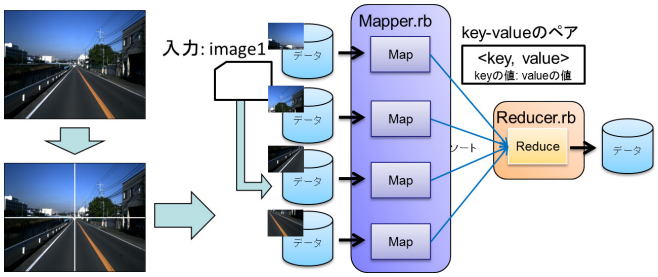
データ全体についての整理された処理結果を取得．

ここで，複数回のMap処理とReduce処理を組み合わせることにより1回のMapReduceでは不可能な複雑な処理も実行可能となる．また，MapReduceでの並列処理を考え，key-value対の割り当てが重要となる．

3.3. Ruby による MapReduce

Hadoop は，元々Java によって作られたものであるため，Hadoop 上の処理をする場合は Java で記述する必要がある[3]. しかし，Hadoop 拡張の Hadoop Streaming を使用すると，標準入出力を介するプログラムを記述するだけで，Ruby 言語を用いて，Map 処理と Reduce 処理を記述できる．そして，動画像処理において MapReduce を用いて大規模データを分散並列処理する

ために，複数の画像を入力として与え，結果として画像処理を行えるかの検証を行う必要性がある．ここで，Hadoop において，Ruby 言語を用いて実行する際には，Hadoop の拡張パッケージである Hadoop Streaming を用いるて実行を行う．また，図 13 において，MapReduce の画像処理の流れと実行文の例を示す．




```
bin/hadoop jar contrib/streaming/hadoop-0.21.0-streaming.jar
-input 入力ファイル -output 出力ファイル
-mapper “ruby パス“ -reducer “ruby パス”
```

図 13　画像処理における MapReduce 処理の流れと Ruby 言語による実行文の例

4. 実験

車載カメラから得られた時系列画像から作成した画像ピラミッドから算出したオプティカルフローより，動きの消失点の座標値が求まる．ここで，消失点が画像中心より右寄りに現れる場合は車載カメラを設置した自動車の進行方向は右であり，消失点が画像中心より左寄りに現れる場合は車載カメラを設置した自動車の進行方向は左である．消失点が画面上に存在しない場合には動きベクトルの偏りより左右どちらに曲がっているのか判断することができる[1]．また，消失点の軌跡を分析することで，自動車の挙動についても検証できる．実験環境は，OS: windows7 Home Premium 32bit, CPU: core i7-860　2.80GHz，メモリ:4GB を用いた．バージョンとしては，Octave3.2.4 である．

また，複数の画像を入力として与え，Map 処理では，画像処理を行い，Reduce 処理では分割されたファイルの順序を保つように統合を行うことで，画像間にまたがる処理が実行可能であるかどうかの検証段階として，グレースケール化を行う．また，画素間にまたがる処理として，特徴量抽出処理を Map 処理で，外部プログラムとして，Ruby 言語から Octave を呼び出し，分割した画像処理を実行し，分割した画像処理結果を収集・結合し，処理結果を出力する．そして，MapReduce の疑似分散を用いて，画像処理が可能かどうかを確かめた．実験環境は，Linux OS Ubuntu 10.10，CPU: Intel PentiumD 2.8GHz(デュアルコア)を用いた．バージョンとしては，Hadoop-0.21.0，Ruby1.8.7，Octave3.2.4 である．ここで，図 14 に特徴量抽出時の MapReduce 処理の Ruby 言語による外部プログラム Octave による出力結果を示す．また，図 15 に Ruby 言語による外部プログラム Octave 起動とプログラムの例を示す．

図 14　画像処理における MapReduce 処理の Ruby 言語による外部プログラム Octave による出力結果画像

```
def octave(filename)
p system("octave",filename)
end
```

```
%test.m
im = imread("test1.jpg");
im = rgb2gray(im);
…
[height width] = size(im);
K=0.04;
for i = 1:height
 for j = 1:width
M = [Ix2(i,j) Ixy(i,j);Ixy(i,j) Iy2(i,j)];
R(i,j) = det(M)-K*(trace(M))^2;
end;end;
Rmax = max(R(:));
Rmin = min(R(:)); R_abs = abs(R);
Rmin = min(R_abs(:));
val = Rmax-Rmin;
for i = 2:height-1
for j = 2:width-1
if R(i,j) > n*Rmax
res(i,j) = 1;
end;end;
…
imwrite(R,"ed_test1-1.jpg");
imwrite(res,"ed_test1-2.jpg");
```

図 15　Ruby 言語による外部プログラム Octave 起動とプログラムの例

5. まとめと今後の課題

本論文では，動画像処理を Lucus-Kanade アルゴリズムを用いておこない，動きベクトルの確認を行い，さらに，動きの消失点推定の実験を行った．また，実際の動きの消失点と推定した動きの消失点との比較を行った．動きの消失点から放射状にオプティカルフローが算出されていることを確認した．また，MapReduce を用いて複数の画像における分散並列処理が実行可能かどうかを検証し，時系列画像の集合である動画像の

理への応用について検討する．

今後の課題としては，提案手法の改良，画像の切り出し領域の検討，時系列画像の大規模実験，MapReduce の実装による性能評価，MapReduce の特性を活かす効率の良い Map 処理や Reduce 処理，そして，key-value 対の検証を行っていきたい．

謝辞

本研究は科研費（22500092）の助成を受けたものである．

参 考 文 献

[1] C. Rasmussen, “Road Compass: following rural roads with vision+ lader using vanishing point tracking,” *Autonomous Robots*, Vol.25, pp. 205-229, 2008.

[2] T. Matsui, T. Imanaka, Y. Kono, “Front Environment Recognition of Personal Vehicle Using the Image Sensor and Acceleration Sensors for Everyday Computing,” *Lecture Notes in Computer Science*, pp. 151-158, 2009.

[3] S Das, Y Sismanis, K S Beyer, R Gemulla, P J Haas, J McPherson, “Ricardo: integrating R and Hadoop,” *Proceedings of the 2010 international conference on Management of data*, 2010.

[4] C. Harris, and M. Stephens, “A combined corner and edge detector,” *Proceedings of the 4th Alvey Vision Conference*, pp.147-151, 1988.

[5] A. Giachetti, M. Campani, and V. Torre, “The Use of Optical Flow for Road Navigation,” *IEEE Transactions on Robotics and Automation*, Vol. 14, no. 1, pp. 34-48, February, 1998.

[6] A. Giachetti, M. Campani, and V. Torre, “The Use of Optical Flow for the autonomous Navigation,” *Proceedings of the 3th European Conference of Computer Vision*, Vol. 1, pp.146-151, June, 1994.

[7] Wilfried Enkelmann, “Obstacle Detection by Evaluation of Optical Flow Fields from Image Sequences,” *Image and Vision Computing*, Vol. 9, no. 3, pp. 160-168, 1991.

[8] 佐藤誠，佐々木宏，“動画像における動きベクトルの階層的推定法，” 電子通信学会論文誌，J69-D, no. 5, pp.771-776, 1986.

[9] J. Dean and S. Ghemawat, “MapReduce:Simplified Data Processing on Large Clusters,” *In Proceedings of Sixth Symposium on Operating System Design and Implementation*, pp. 137-150, December, 2004.

[10] 布施孝志，清水英範，堤盛人，“オプティカルフロー推定における光学勾配法の比較分析，” 応用測量論文集，Vol.11, pp.45-52, 2000.

[11] 木村和広，太田直哉，金谷健一，“精密なノイズモデルによるオプティカルフローの検出，” 情報処理学会研究報告，96-CV-99-6, pp. 37-42, 1996.

[12] 市原直彦，井上博人，藤田隆二郎，伊藤宏平，安士光男，”車載カメラ応用技術～画像認識カーナビゲーションと車載ロボット～，” PIONEER R&D, Vol. 16, no. 1, 2009.

[13] 藤原良子，山内仁，高橋浩光，“車載カメラによる動画像からの走行軌跡追跡抽出に関する一検討，” 電子情報通信学会技術研究報告. IEICE，画像工学，Vol. 103, no. 451, pp. 5-10, 2003.