日本ソフトウェア科学会第 23 回大会（2006 年度）

blog 検索エンジンもぶろげっと™への動画検索機能の導入

Video search Introduced into blog search engine Mobloget™

高地 利幸　林 貴宏　尾内 理紀夫
Toshiyuki TAKACHI, Takahiro HAYASHI, Rikio ONAI

電気通信大学
The University of Electro-Communications

blog に付随する動画に注目し，動画を含む blog 記事を収集・解析して，それらを検索するシステム「もぶろげっと動画版」を試作した．本システムは，既存システムである画像付き blog を対象とした検索エンジン「もぶろげっと」を改良し，blog 記事からの動画の抽出機能や，動画からのサムネイル画像作成機能を追加した．本システムの検索結果には，blog 記事のタイトル，要約文と共に，動画から作成したサムネイル画像が表示される．マウスポインタをサムネイル画像上に移動させると，カット点検出を用いて作成された 10 枚のサムネイル画像がページの切り替えなしに表示される．サムネイル画像作成においては，単一色画像や砂嵐画像などのサムネイル画像として適していない画像を選択しない方法を提案，実装，評価した．

1 はじめに

現在，様々な種類の blog が普及してきている．テキストのみで構成されている blog の他にも，画像を主体とした「moblog」や「photolog」，音声や音楽付きの「Podcasting」，動画付きの「vlog(ブイログ)」といったものがある．この中で，今後ますます普及していくと予想されるのが vlog である．vlog とは video blog のことで，ショートビデオを掲載した blog のことを指し，movie blog，動画 blog などとも呼ぶ．最近アメリカでこの vlog が流行の兆しを見せ，ROCKETBOOM[1]などのサイトが話題になっている．日本でも携帯電話などで撮った動画を，写真と同様に簡単に blog に取り込めるサービスが goo ブログ[2]などで始まっている．この vlog が普及すると共に，「動画も参考にして blog を検索したい」，「blog に付随している動画を検索したい」というような要求が高まることが予想される．

そこで本研究では，著者らの研究室で開発した画像を含む blog 記事を検索するシステム「もぶろげっと」[1]を拡張させ，動画を含む blog 記事を検索できるようにする．

以下，2 章でもぶろげっとのシステムの概要を述べ，3 章で新たに提案するもぶろげっと動画版について述べる．4 章で blog 記事から動画を抽出する処理について述べ，5 章で動画からサムネイル画像を作成する方法について述べる．6 章で，5.2 章で述べる方法の評価実験について説明し，その結果と考察を述べる．7 章で関連研究について述べ，8 章で本稿をまとめる．

2 もぶろげっとのシステム概要

もぶろげっと動画版を作成するにあたって，基になるもぶろげっとのシステムの概要について説明する．

もぶろげっとは，大きく分けて収集・解析部と検索部から構成される．以下 2.1 で収集・解析部について説明し，2.2 で検索部について説明する．

2.1 収集・解析部

blog を構成する Web ページを，本稿では blog ページと呼ぶことにする．blog ページには，

(1) トップページ(最近投稿された blog 記事をまとめたページ)
(2) カテゴリごとに blog 記事をまとめたページ群
(3) 月ごとに blog 記事をまとめたページ群
(4) blog 記事を 1 つだけ含むページ群(1 投稿ご

[1] ROCKETBOOM, http://www.rocketboom.com/vlog/
[2] goo ブログ，http://blog.goo.ne.jp/index.php

とに blog 記事を分けたページ群)

などがある．なお，本稿で「blog 記事」とは，ユーザによって投稿されたテキスト，画像，動画を指す．

収集・解析部では，Web から(4)に分類されるページを収集し，blog 記事や画像の抽出などの解析を行う．収集・解析部は以下の 4 つのモジュールで構成される．

1. **blog 発見モジュール**：クローリング対象となる blog のトップページの URL を Web から収集し，データベースに登録する．
2. **クローラ**：データベースに登録されたページをクローリングし，ページに含まれるリンクをたどり，上記(4)に分類される blog ページを収集する．
3. **blog 記事抽出モジュール**：クローラによって収集された blog ページから，blog 記事の部分を抽出する．すなわち，blog ページ内のナビゲーション用のリンクや広告などの部分は抽出しない．
4. **画像抽出・メタデータ作成モジュール**：blog 記事抽出モジュールで抽出された blog 記事から画像を抽出する．抽出した画像から検索部で使用するための，サムネイル画像を作成する．画像が含まれていない blog 記事は，データベースに登録しない．また，抽出した画像に対して，画像の URL，データ形式，縦横の大きさ，前方後方のテキスト，alt 属性を抽出し，画像のメタデータを作成する．

2.2 検索部

収集・解析部によって抽出された blog 記事に対する全文検索の機能を，ユーザに提供する．検索部はインタフェースと検索モジュールで構成される．

1. **インタフェース**：ユーザからクエリを受け取り，検索結果を提示する．各 blog 記事のタイトル，要約，サムネイル画像を表示する「通常モード」と，サムネイル画像のみを並べて表示する「画像だけモード」がある．
2. **検索モジュール**：インタフェースからクエリを受け取り，データベースに蓄積された blog 記事に対して全文検索を行い，その結果をインタフェースに返す．インデクシングには Namazu[3]を用いている．

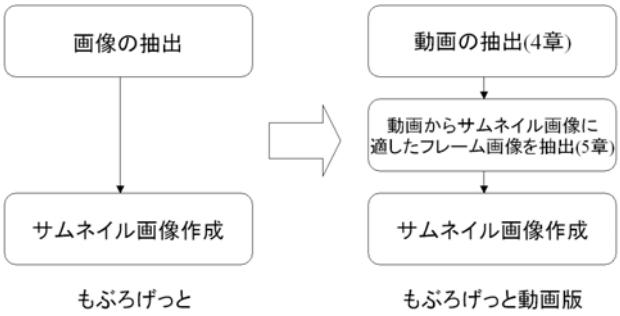

図 1　画像抽出・メタデータ作成モジュールの変更

3 もぶろげっと動画版の試作

もぶろげっとの収集・解析部における画像に対する処理を，動画に対する処理に変更したもぶろげっと動画版を試作した．収集・解析部の中でも，特に画像抽出・メタデータ作成モジュールにおいて，図 1 に示すような変更をすることで本システムを実装した．もぶろげっとでは，画像を抽出し，サムネイル画像を作成する．それに対し動画版では，動画を抽出し，動画からサムネイル画像に適しているフレーム画像を抽出して，サムネイル画像を作成する．

本システムは現在 http://video.mobloget.jp/ にて試験運転中である．本システムの特徴を以下に示す．

- 動画付きの blog 記事を検索可能．
- 現在対象としている blog サービスは，livedoor blog と so-net blog．
- 対応している動画フォーマットは，AVI，WMV，MOV(QuickTIme)，MPEG，RM(RealVideo)，MP4 の 6 種類．
- インタフェースには，検索結果として，各 blog 記事のタイトル，要約文，動画から作成したサムネイル画像を表示する「通常モード」(図 2)と，サムネイル画像のみを並べて表示する「サムネイル画像だけモード」がある．

[3] Namazu，http://www.namazu.org/

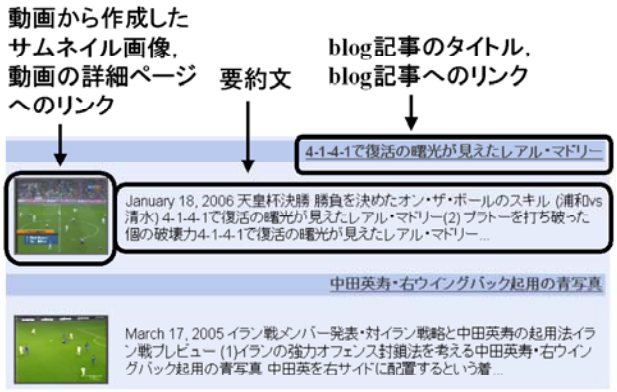

図 2　本システムの実行画面（通常モード）

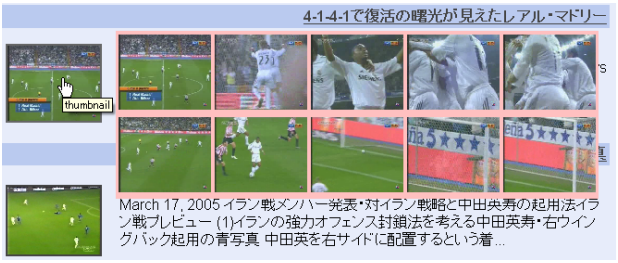

図 3　10 枚のサムネイル画像

図 4　詳細ページ

- blog 記事のタイトルをクリックすると，その記事へ移動できる．
- ユーザがマウスポインタをサムネイル画像上に移動させると，JavaScript によってページの切り替えなしに 10 枚のサムネイル画像が表示される(図 3)．また，ユーザがマウスポインタをサムネイル画像から離すと 10 枚のサムネイル画像は消去される．
- サムネイル画像をクリックすると動画の詳細ページが表示される．動画の詳細ページには，動画のフォーマット，再生時間，ファイルサイズ，画面サイズ，動画へのリンク，10 枚のサムネイル画像が表示されている(図 4)．

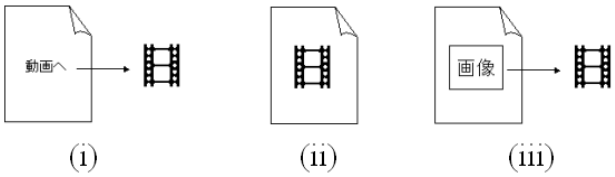

図 5　動画のリンクパターン

4 blog 記事からの動画抽出

本システムで，抽出対象とする動画について 4.1 で説明し，動画抽出における問題点について 4.2 で説明する．

4.1 抽出対象とする動画

収集した blog 記事から，動画の URL を抽出する．本システムでは，図 5 に示す 3 つのパターンでリンクされた動画のファイルを，その blog 記事に含まれるものとし，抽出する．(i)は a タグを用いて，テキストに動画ファイルへのリンクを貼るパターンである．(ii)は embed タグを用いて直接動画ファイルを埋め込むパターンである．(iii)は img タグで埋め込まれた画像に，a タグを用いて動画ファイルへのリンクを貼るパターンである．

なお，テキストに動画の URL が書いてあるが，リンクが貼られていない(a タグが使用されていない)パターンがある．本システムでは，このようなパターンで参照されている動画は，blog 記事に含まれる動画ではないとして，抽出しない．

4.2 動画抽出の問題点

本システムが，現時点で抽出できない動画には以下のようなものがある．

1. embed タグではなく，object タグを用いて動画を埋め込んでいる場合
2. 動画が zip ファイルなどに圧縮されている場合
3. 動画へのリンクが切れている場合(404 エラー)
4. ファイルの拡張子が間違っている場合
5. 動画へのリンクがメタファイル内から貼られている場合
6. 未対応のコーデックが使われている場合

1，2 は，今後対処していきたい．3 の場合については，現在対処することは考えていない．4 の場合については，現在動画フォーマットの判定に，

拡張子の情報しか用いていないことが原因である．したがって，拡張子のみでフォーマットを判定するのではなく，ファイルの中身を見て，フォーマットを判定するような処理が必要である．5 の場合については，現在動画のメタファイルに対する処理をしていないことが原因である．動画のメタファイルとは，動画ファイル本体の URL，再生方法，著作権，タイトルなどが記述されたテキストファイルである．よって，blog 記事内でメタファイルへのリンクを発見した場合は，そのメタファイルの中に記されている動画ファイル本体の URL を抽出して，動画をダウンロードするという処理が必要である．6 の場合については，定期的にコーデックを入手するなどして対処していきたい．

5 動画からのサムネイル画像作成法

5.1 処理の流れ

抽出した動画からフレーム画像を切り出し，サムネイル画像を作成する．サムネイル画像の作成法としては，動画の冒頭フレームの画像を選択し，これをサムネイル画像とする方法が考えられる．しかし動画は，真っ黒や真っ白などの単一色の画像や，砂嵐画像で始まることがある．したがって，動画の冒頭フレームの画像を何も考えずに選択したのでは，動画の内容とは関係のないサムネイル画像になってしまうことがある．サムネイル画像は検索結果をブラウジングする際に重要なものであり，そのような画像は適していない．また，1 枚のサムネイル画像だけで動画の内容を表すのは難しい．そこで，動画の各ショットの先頭フレーム画像もサムネイル画像として取り出す．検索結果で複数のサムネイル画像を表示することで，動画の内容を理解しやすくなると思われる．

フレーム画像を切り出す処理は，AVI，WMV，MPEG，RM は Direct Show[4]で行い，MOV，MP4 は QuickTime SDK[5]で行っている．全体の処理の流れは以下のようになる．

まず，動画冒頭の不要区間を除去する(5.2 章)．なお本稿では，動画冒頭に出現することがある単一色や砂嵐などの区間を，サムネイル画像を作成するには適していない区間とみなし，不要区間と呼ぶことにする．不要区間を除去して残った区間に対して，カット点検出(分割 χ^2 検定法)[2]を用いて，各フレーム画像間で分割 χ^2 検定法の評価値を求める．この評価値が高いと，フレーム画像間での変化が大きいとしてカット点(ショットの始まりのフレーム)を検出できる．この評価値の高いフレーム画像 10 枚から，10 枚のサムネイル画像を作成する．

5.2 不要区間除去

不要区間を除去するために，動画のフレーム画像を先頭から順に見ていき，サムネイル画像に適している最初のフレーム画像を発見する．そのフレーム画像より前の区間を不要区間として除去する．なお本稿において，「サムネイル画像に適していない画像」とは，単一色画像や砂嵐画像のような動画の内容とは関係のない画像であることを表し，「サムネイル画像に適している画像」とは，動画の内容と関係のある画像であることを表すこととする．

動画冒頭のフレーム画像がサムネイル画像に適しているかどうか判定する方法を以下に示す．なお，これらの画像処理は OpenCV[6]の関数を使用して実装した．

1. 動画の最初のフレーム画像を切り出す．
2. 切り出したフレーム画像をグレースケール画像に変換する．
3. グレースケール画像にメディアンフィルタをかけ，雑音を除去する．
4. メディアンフィルタをかけた画像に，Canny フィルタ[3]をかけて，エッジを検出する．
5. Canny フィルタをかけた画像に，Hough 変換を用いて線分検出をする．検出した線分の数に対して閾値を設定し，閾値を超えたフレーム画像は，サムネイル画像に適していると判定し，超えなければ適していないと判定する．現在は，閾値を経験的に 20 と

[4] DirectShow(DirectX), http://www.microsoft.com/japan/msdn/directx/
[5] QuickTimeSDK, http://developer.apple.com/quicktime/
[6] OpenCV：画像処理用 C ライブラリ http://www.intel.com/research/mrl/research/opencv/

している．

6. 5 でサムネイル画像に適していると判定されれば処理を終了する．適していないと判定された場合は，1 秒後のフレーム画像を切り出し，再び 2 へ戻る．

3 のメディアンフィルタをかける処理は，主に砂嵐画像をサムネイル画像に適していないと判定する目的で行っている．砂嵐画像は，メディアンフィルタをかけ全体をぼかすことで，エッジが検出されにくくなる．その結果，5 の Hough 変換において，検出される線分数が少なくなり，サムネイル画像に適していないと判定することができる．

5 で Hough 変換を用いて線分検出をするのは，まず 1 つ目に，単一色画像や図 6 に示すようなカラーバー画像をサムネイル画像に適していないと判定する目的がある．単一色画像はエッジがないので，線分数は 0 になる．カラーバー画像も，直線のみで構成されているので，線分数はそれほど大きな数にはならない．したがって，これらの画像をサムネイルに適していないと判定することができる．また 2 つ目として，動画の冒頭に字幕がある場合に，その字幕が表示されているフレーム画像を，サムネイル画像に適していると判定する目的がある．ここでの字幕とは，動画の冒頭に表示され，その動画の題名や，内容を表している文字のことを指す．字幕は通常，文字が読みやすいように，背景色と文字色のコントラストが高いものである．したがって，エッジが明確に検出される．さらに文字は，線分が多数検出される傾向がある．これにより，字幕画像をサムネイル画像に適していると，判定することができる．

メディアンフィルタをかけた場合と，かけなかった場合の，砂嵐画像に対する適用例を図 7 および図 8 に示す．なお図中の矢印の数字は，上で述べた処理の数字と対応している．つまり，2 はグレースケール変換，3 はメディアンフィルタ，4 は Canny フィルタ，5 は線分検出である．図 7 のようにメディアンフィルタをかけた場合は，エッジがほとんど検出されていないのに対し，図 8 のようにメディアンフィルタをかけない場合は，エッジが多数検出されていることが確認できる．さらに，字幕画像に対する適用例を図 9 に示す．文字の部分において線分が多数検出できていることが確認できる．

図 6　SMPTE カラーバー画像[7]

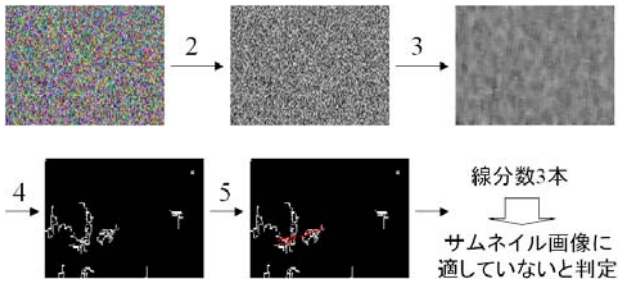

図 7　砂嵐画像に対する適用例

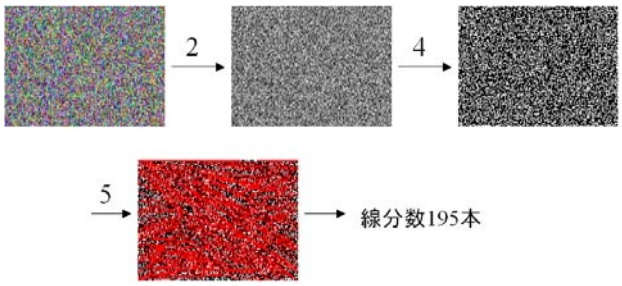

図 8　2 の処理なしでの砂嵐画像に対する適用例

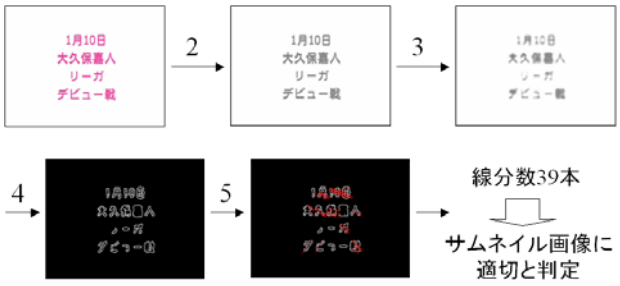

図 9　字幕画像に対する適用例

6 評価実験

6.1 実験方法

5.2 章で述べたサムネイル画像に適している最初のフレーム画像を作成する方法によって，システムが選ぶサムネイル画像と，人が選ぶサムネイル画像とを比較するために以下のような実験を行った．

まず，動画冒頭から 1 秒毎のフレーム画像を順

[7] http://www.smpte.org/

に並べた図 10 のような実験データを用意する．そして，被験者に動画が開始したと思われるフレーム画像を選択してもらう．その結果と，5.2 章の方法によって得られたフレーム画像とを比較し

図 10　実験データ例

図 11　失敗例

た. 動画は 30 本, 被験者は 20 人で実験を行った.

6.2 結果と考察

図 10 の場合は，被験者の多くが t=1 のフレーム画像を選択し，システムも t=0 の単一色画像をサムネイル画像に適していないと判定し，t=1 のフレーム画像がサムネイル画像に適していると判定した．また動画冒頭にカラーバー画像，砂嵐画像がある場合も，被験者・システム共に，それらを選択せずにサムネイル画像に適していると思われるフレーム画像を選択した.

一方で図 11 は，被験者の多くは t=1 のフレーム画像を選択したが，システムでは t=0 から t=4 は, サムネイル画像に適していないと判定し, t=5 のフレーム画像を適しているとして選択した．t=1 から t=3 のフレーム画像をサムネイル画像に適していないと判定した理由は，文字数が少ないために検出した線分数が閾値 20 を超えなかったためである．さらに t=4 のフレーム画像を適していないと判定した理由は，文字が出始めで薄かったために，エッジがあまり検出されず，そのために線分数が少なくなってしまったためである．このように，現在の方法では文字が小さかったり，文字数が少なかったり，コントラストが低くてエッジが検出されにくかったりする場合において，サムネイル画像に適している画像であっても，適していないと判定してしまうことがある.

以上から，単一色画像，カラーバー画像，砂嵐画像などを，高い確率でサムネイル画像に適していないと判定する一方で，適している画像も選択できないことがあることが確認できた．今後，サムネイル画像に適している画像を正しくサムネイル画像に選択するように，改良が必要であると思われる.

7 関連研究

既存の blog 検索エンジンは多数あるが, 本システムのように, 動画の付いている blog 記事を検索できるシステムは，現在見当たらない.

また，blog を対象にしたものではなく，Web 全体を対象とした動画の検索エンジンには，Yahoo! の動画検索[8]や goo 画像・動画・音楽検索[9]などがある．さらに，ニュースやスポーツ情報などを動画で配信・検索できる MSN ビデオ[10]や，米テレビ番組，ユーザからアップロードされた動画を販売・検索できる Google Video[11]など様々な動画検索システムがある．このようなシステムで検索できる動画は，企業などが作る宣伝や，商品の CM などが上位にランキングされる傾向がある．そのため，個人の動画が下位になり検索しずらい．それに比べて, 本システムは対象を blog に限定することにより，個人が作る生活に密着した動画のみを検索することができる.

8 おわりに

blog 検索エンジンもぶろげっとに，動画に対する処理機能を追加して，もぶろげっと動画版を試作した．サムネイル画像作成においては，不要区間を除去した後，カット点検出を用いて 10 枚のサムネイル画像を作成する方法を提案した．また，不要区間を除去するために，単一色画像や砂嵐画像などをサムネイル画像として選択しない方法を提案，実装，評価した．評価実験から，単一色画像やカラ

---

[8] Yahoo!動画検索, http://search.yahoo.co.jp/video?&ei=UTF-8&p=
[9] goo 画像・動画・音楽検索, http://bsearch.goo.ne.jp/
[10] MSN ビデオ, http://jp.video.msn.com/v/ja-jp/v.htm
[11] Google Video, http://video.google.com/

ーバー画像や砂嵐画像などを高い確率でサムネイル画像に適していないと判定する一方で，適している画像が適していないと判定する場合があった．

今後の課題としては，サムネイル画像作成法において，サムネイル画像として適している画像を，適していないと判定してしまう場合を改善する方法の検討がある．また，試作したもぶろげっとの動画検索の検索精度や，ユーザビリティなどを評価する予定である．

参考文献

[1] 井原伸介,林貴宏尾,内理紀夫 ：”画像情報を含むblog記事検索システムの開発”，電子情報通信学会論文誌 D,Vol.J89-D No.6 pp.1236-1247, 2006

[2] 長坂,田中 ：”カラービデオ映像における自動索引付け法と物体探索法”，情報処理学会論文誌 Vol.33, No.4, pp.543-550, 1992

[3] Canny,J.：“A Computational Approach for Edge Detection” , IEEE Trans. PAMI, vol.8, No.6, pp.679-698, 1986