Development of VCAPG for NetworkCAMera

はじめに

常時接続、ブロードバンドの普及にともない、ネットワークカメラを使った遠隔監視システムの導入が容易になりつつある。これらを活用しネットワークを介しコンピュータに画像を取り込み解析できると、セキュリティ・介護などに有効活用できると考えられる。

ここでは、ネットワークカメラを活用するための MATLAB 関数を作成することを目的とする。

ここで取り上げるネットワークカメラは、Panasonic 社製のものであるが、ネットワークを介し操作するための CGI(Common Gateway Interface)が公開されている。

これらは、CGI は、KX-HCM1, KX-HCM2,KX-HCM130,KX-HCM170,KX-HCM180 などいろいろなタイプのカメラに共通である。ここでは、KX-HCM1 を使い、どのようにしてリアルタイムで動画像データをネットワークを介し取り込み処理するのかを述べる。

MATLAB、MATLAB-Java、C-mex プログラミングなどを駆使し、リアルタイムでカメラ制御、それに画像処理可能な環境の構築を実現する。そのため MATLAB へのネットワークカメラ画像の取り込みプログラムの開発、カメラ制御プログラムの作成を行う。

これらの開発過程で作成したサンプルプログラムを交えながら解説を行っていく。

telnet による HTTP プロトコル応答

HTTP プロトコルとは、HyperText Transfer Protocol の略で、インターネットでホームページなどをブラウジングするときに利用するプロトコルである。HTTP プロトコルは TCP/IP のプロトコルで、通常 80 番ポートを使ってアクセスする。基本的には、メッセージを要求しその応答結果(レスポンス)を表示するといった機能を持っている。では、実際に WEB サーバがどのような応答をしているのかチェックしてみよう。簡単な方法は、コマンドプロンプト上での telnet コマンドによる確認である。コマンドプロンプトを起動し、telnet コマンドを入力する。GET コマンドは、HTTP プロトコルのコマンドのひとつである。この例では、インターネットにあるホームページを telnet コマンドにより見る例を示す。

```
telnet www.ikko.k.hosei.ac.jp 80
GET / HTTP/1.1
```

GET・・・は、telnet 画面上では、見えないかもしれないが、そのまま入力し、enter キーを 2 回押すと完了する。データが入力し終わると、以下のような表示が出て、終了する。

```
HTTP/1.1 200 OK
Date: Tue, 09 Sep 2003 01:05:08 GMT
Server: Apache
```

```
Accept-Ranges: bytes
Content-Length: 9615
Connection: close
Content-Type: text/html
Content-Language: ja

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="ja">
・・・
```

DOCTYPE 以降は、通常見る HTML ファイルであるが、HTTP プロトコルでは、通信した時にその時間、サーバ名、コンテンツの長さ、タイプ、言語などの情報を付加し、ブラウサに応答を返している。このように Web ブラウサだけでは、ファイル内容しか見えないが、HTTP プロトコルの通信では、それ以外の情報、時間、サーバー名やデータの大きさ言語などの情報も送っていることがわかる。

MATLAB 上での Java を使った、プロトコルチェックスクリプトの作成

では、telnet コマンドで行ったことと同様のことを MATLAB-Java で行うにはどのようにすればよいのであろうか？ここでは、MATLAB-Java を用い対象となるネットワークカメラとの通信を試みてみる。そのための設定の仮定として、KX-HCM1 を用い、内部の IP アドレスは、192.168.1.3 を設定した例で行う。そのためのスクリプトを以下に示す。Java では、URL クラスを使えば、http プロトコルについて気にせずに通信することができるが、ここではあえて、HTTP プロトコル通信内容を見るために Socket クラスにより通信を実現している。

mclient.m

```
PORT = 80;
HOST='192.168.1.3';
aSocket=java.net.Socket(HOST,PORT);
autoflush = 1;
out     = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
aReader = java.io.BufferedReader(java.io.InputStreamReader(aSocket.getInputStream));
out.println('GET / HTTP/1.1');
out.println('Host: localhost:8080');
out.println('Connection: Close');
out.println('');
loop = 1;
cnt = 0;
while (loop)
    s = readLine(aReader);
    if isempty(s)
        cnt = cnt + 1;
    else cnt = 0;
    end
    if(cnt > 3) break;end
    fprintf('%s¥n',char(s));
end
close(aSocket);
```

一般に、ネットワークカメラは、インターネットに接続していると、ネット上の不特定多数の人から見られてしまうため、セキュリティ確保のためパスワード設定をしている場合が多い。

パスワードが設定されている場合には、HTTP プロトコルは、どのような応答を返すのであろうか？また、そのパスワードを解除するにはどのように設定すればよいのであろうか？

パスワードを設定した状態で、実際にネットワークカメラにスクリプトを使いアクセスした例を示す。

```
>> mclient
HTTP/1.0 401 Unauthorized
Server:U S Software Web Server
```

```
WWW-Authenticate:Basic realm="General User"

<HTML>
以下続く
</HTML>
```

得られた情報の HTTP/1.0　401 の後を見ると、Unauthorized という文字が出ている。これは、ページ認証ができていないというエラー表示である。またプロトコルの下のほうを見てみると、WWW-Authenticate:Basic realm="General User"と出力されている箇所がある。これは、認証に Web サーバでよく使われる BASIC 認証を使っていることを示している。

Web サーバにおける BASIC 認証

BASIC 認証では、HTTP プロトコルで必要な GET などのヘッダコマンド以外に、Authorization ヘッダが必要となる。Web サーバ側では、Authorization：ヘッダに記述されている ID とパスワードを解析し、その ID とパスワードがサーバー側に登録されているものと同じものであれば、アクセスを許可し、そうでない場合にはアクセスを許可しないという制御を行っている。一般に、BASIC 認証でアクセス制限されたサイトにアクセスするときの流れは、次のとおりである。

１．ユーザがブラウサに目的の URL を入力する。
２．ブラウサ内部では、GET　コマンドによりコンテンツを要求する。
３．サーバでは、HTTP/1.0 401 Unauthorized のレスポンスを返し、ブラウサに認証が必要であることを伝える。
４．ブラウサ内部では、ポップアップウィンドウを出して、ID とパスワードの入力を求める。
５．今度は、ブラウサは、Authorization:ヘッダに ID とパスワードを付加し、GET コマンドにより再度、コンテンツを要求する。
６．サーバが ID とパスワードを解析して認証 OK であれば、コンテンツをブラウサに返す。

このときの Autorization:ヘッダには、BASIC 認証の ID とパスワードとを：（半角のコロン）でつなげた文字列を、base64 により符号化した文字列が格納されている。

BASE64 とは？
base64 とは、3 バイトのデータ（バイナリデータを含む）を 4 バイトのテキストデータに変換するエンコード方式である。例えば 3 バイトのバイナリデータがあるとしたとき、3x8=24 ビットで表される。これを 6 ビットごとの 4 つのデータに区切り、区切られた 4 つのデータ(6 ビット)の上位 2 ビットを 00 とみなすと、4 バイトのデータをみなすことができる。各バイトは上位ビットが 0 のため 4 バイトはテキスト文字列として表現できる。6 ビットのデータで表現するため、2 の 6 乗は 64 となることから BASE64 の 64 はこの数字からきている。
つまり、3 バイトで表現されたバイナリデータを１つのまとまりとし 4 文字のテキストデータ変換する方法である。
これは、ID やパスワードを符号化（暗号化といっても解読が簡単にされてしまうので）に使う以外にも電子メールなどバイナリデータが送れない環境で、バイナリデータを転送したい場合 base64 変換によりテキストデータに変換しやり取りするなどに使用する場合が多い。

%　3 バイトのデータを 4 つに分割する。
%　aaaaaabb bbbbcccc ccdddddd
% から
%　00aaaaaa 00bbbbbb 00cccccc 00dddddd
となっている。できた、4 つの最大６ビットデータ（0～63）をそれぞれ ASCII 文字にマッピングすることでテキストデータに変換する。

```
%     0   - 25   ->   A-Z
%     26 - 51   ->   a-z
%     52 - 61   ->   0-9
%     62          ->   +
%     63          ->   /
```

これを計算すればよい。

例題：

たとえば、01234:567890 な文字があった場合、これを base64 でエンコードしてみよう。

uint8 関数により文字列を 10 進数の ASCII コードに変換する。

```
>> aaa=uint8('01234:567890')
aaa =
     48      49      50      51      52      58      53      54      55      56      57      48
```

reshape 関数により 3 文字づつ分ける。なお、MATLAB では行列での数字の順序が縦方向になっているので C 言語などで実装するときには注意が必要である。

```
ccc=reshape(aaa,3,4)
ccc =
     48      51      53      56
     49      52      54      57
     50      58      55      48
```

次に先頭の文字 ccc(1,:)を bitshift 関数により 2 ビットシフトする。

```
%      aaaaaabb-> 00aaaaaa
>> ccc(1,:)
ans =
     48      51      53      56
>> aaaaaa=bitshift(ccc(1,:),-2)
aaaaaa =
     12      12      13      14
```

これで最初の 1 バイト分ができる。

```
%      aaaaaabb bbbbcccc ->00bbbbbb
```

次の２バイト目は、２つの文字にまたがっているのでちょっと厄介である。

先頭文字をまず、bitand 関数により 3 でマスクをかける。これは、下位２ビットだけを残し、上位 6 ビットにゼロ代入することに相当する。次に、bitshift 関数により上位に 4 ビットシフトし、次に、２番目の文字の数値を 4 ビット下位にシフトし、bitor により全体をつなげている。

```
>> bbbbbb=bitor(bitshift(bitand(ccc(1,:), 3), 4), bitshift(ccc(2,:), -4))
bbbbbb =
      3      51      19       3
```

次の 3 番目の文字も２番目の文字と同様の操作を行う。

```
%      bbbbcccc ccdddddd -> 00cccccc
>>cccccc=bitor(bitshift(bitand(ccc(2,:), 15), 2), bitshift(ccc(3,:), -6))
cccccc =
      4      16      24      36
```

最後の 4 番目は、6 桁にマスクをかけ、上位 2 桁をゼロにする。

```
%     ccdddddd    -> 00dddddd
>> dddddd=bitand(ccc(3,:), 63)
dddddd =
     50      58      55      48
```

計算した結果をまとめると、

```
>>ddd=[aaaaaa;bbbbbb;cccccc;dddddd]
ddd =
```

```
    12    12    13    14
     3    51    19     3
     4    16    24    36
    50    58    55    48
```

次に、コロン演算子によりベクトルに変換、転置により横ベクトルに変換する。

```
>> ddd=ddd(:)'
ddd =
    12   3   4 50 12   51   16    58    13    19    24    55    14     3    36    48
```

あとは、これら数値にテキストコードをマッピングすればよい。MATLAB では、ベクトルで計算ができるため、以下のようにすればたったの 6 行で全てのコードを変換することができる。

```
eee=ddd;
i =              (ddd <= 25); eee(i) = 'A'          + double(ddd(i));
i =(26 <= ddd)&(ddd<= 51); eee(i) = 'a' - 26 + double(ddd(i));
i =(52 <= ddd)&(ddd<= 61); eee(i) = '0' - 52 + double(ddd(i));
i =                ddd== 62; eee(i) = '+';
i =                ddd== 63; eee(i) = '/';
```

この計算により計算された、eee がその結果となる。
また、char 関数を使うことで、ASCII コードから文字列に変換することができる。

```
>> eee
eee =
    77    68    69   121    77   122    81    54    78    84    89    51    79    68
107   119
>> char(eee)
ans =
MDEyMzQ6NTY3ODkw
```

これが変換された文字列となる。
なお、変換する文字列の長さが 3 の倍数である場合には問題ないが、3 の倍数で無い場合には、0 を挿入する必要がある。符号化される前の文字列には、'='を挿入する。
これらの手順を関数 M ファイルとして作ればよい。

課題

1．BASE64 により自分の E-mail アドレスを符号化せよ。
2．符号化された" SG9zZWlVbml2LlN5c3RlbXMgYW5kIENvbnRyb2w="を逆変換せよ。
3．任意の長さのデータを入力しても BASE64 により変換できる base64encode.m を作成せよ。
4．base64encode により符号化されたデータを逆変換する base64decode.m を作成せよ。

一度、ID とパスワードがわかれば常に、Autorizaiton タグに BASE64 により符号化した ID とパスワードを送信することで、直接 BASIC 認証を通過することができる。
そのための Java プログラム mclient1.m を以下に示す。ID は、01234、パスワードは、567890 であるとすると、BASE64 で符号化した文字列は、PASS=' MDEyMzQ6NTY3ODkw';となるので、

```
PORT = 80;
HOST='192.168.1.3';
PASS='MDEyMzQ6NTY3ODkw';
aSocket=java.net.Socket(HOST,PORT);
os     = aSocket.getOutputStream;
autoflush = 1;
out      = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
aReader = java.io.BufferedReader(java.io.InputStreamReader(getInputStream(aSocket) ));
out.println('GET / HTTP/1.1');
aaa=['Authorization: Basic ',PASS];
out.println(aaa);
out.println('Host: localhost:8080');
```

```
out.println('Connection: Close');
out.println('');
loop = 1;
cnt = 0;
while (loop)
    s = readLine(aReader);
    if isempty(s)
        cnt = cnt + 1;
    else cnt = 0;
    end
    if(cnt > 3) break;end
    fprintf('%s¥n',char(s));
end
close(aSocket);
```

このプログラムを実行した場合の HTTP プロトコルの HTTP ヘッダを見ると、認証を通過し、OK の表示がされていることが確認できる。

```
>> mclient1
HTTP/1.0 200 OK
Server: U S Software Web Server
Connection: close
Cache-Control: must-revalidate = no-cache
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
   <TITLE>Network Camera </TITLE>
   <META HTTP-EQUIV="expires" CONTENT="0">
   <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
   <META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
   <META HTTP-EQUIV="Refresh" CONTENT="1;URL=Top">
</HEAD>
<BODY>
</BODY>
</HTML>
```

Authorization タグに符号化した ID とパスワードを設定することにより、認証を通過する。
Authorization タグ以降には、頻繁にキャッシュについて無効にする設定がなされている。たとえば、Cache-Control タグ、HTML タグ内の META タグでも、Cache-Control である。これは、ネットワークカメラのようなアプリケーションでは、キャッシュが有効になってしまうとデータの更新が反映されないための指定である。

STEP3　MATLAB-Java を使ったカメラ制御

認証が通過できれば、GET 以降に適当なコマンドを設定することでカメラの制御などが可能になる。詳細コマンドについては、Panasonic ネットワークカメラ　技術参考資料　ネットワークカメラ CGI 利用説明書に記述してあるので参考にしてもらいたい。
では、まず、簡単なカメラ制御のスクリプト例を紹介する。ネットワークカメラのパンの左右移動、チルトの上下移動には、nphControlCamera コマンドを用いる。
つまり、HTTP プロトコルの GET コマンド

```
GET / HTTP/1.1
```

のところを変更し、

```
GET /nphControlCamera?Direction=PanLeft　HTTP/1.1
```

と変更し、送信すれば、左方向にカメラが動く。

これは余談になるが、Web ブラウサでポップアップの認証画面をなしでアクセスするには、
以下のようにする。

```
http://01234:567890@192.168.1.3/nphControlCamera?Direction=PanLeft
```

なおこの場合、ID：01234 とパスワード:567890 の場合である。ID とパスワードの間の区切りは、コロンであり、パスワードの URL との区切りには@を使う。
この方法は、Web ブラウサが対応している場合の機能であり、MATLAB-Java 上で動作させるためには通用しない。
そこで、この場合のコマンドと等価な役割をする MATLAB スクリプトを示す。

mclient2.m

```
PORT = 80;
HOST=”192.168.1.3;
PASS=’ MDEyMzQ6NTY3ODkw';
aSocket=java.net.Socket(HOST,PORT);
autoflush = 1;
out     = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
motion=[{'PanLeft'},{'PanRight'},{'TiltUp'},{'TiltDown'}];
cmd = 1;
aaa=['GET /nphControlCamera?Direction=',motion{cmd},' HTTP/1.0'];
out.println(aaa);
aaa=['Authorization: Basic ',PASS];
out.println(aaa);
out.println('Host: localhost:8080');
out.println('Connection: Close');
out.println('');
close(aSocket);
```

このスクリプトの cmd の値を、1 から 4 まで変更することで、左右、上下の移動が実現できる。このカメラ制御の仕様説明書によると、カメラの位置制御は、角度指定など絶対角度指定はできない。そのため、角度制御したい場合には、まず、カメラをホームポジションに移動した後、何回移動したのか把握しておく必要がある。

課題　HCM1 では、左右 20 ステップで、±60 度、上下 16 ステップで－45 度～0 度の操作が可能になっている。今、カメラがホームポジションパン（右から）－60 度、チルト（上から）0 度にあると仮定する。この時、任意の角度　パン（-60～60 度まで)、チルト（－45 度～0 度）を入力した時、その向きに向くための命令を生成する関数を作成せよ。

pan= -60+120/20:120/20:60;
tilt= (-45-45/16):45/16:0;

Example
clear sample;% イニシャライズ
 sample(3,3); % pan(3), tilt(3) の順、つまり－54 度、チルトを－42 度にする。
    nphControlCamera?Direction=PanLeft
    nphControlCamera?Direction=PanLeft
    nphControlCamera?Direction=PanLeft
    nphControlCamera?Direction=TiltDown
    nphControlCamera?Direction=TiltDown
    nphControlCamera?Direction=TiltDown
と言うコードを表示し、次にもう一度
     sample(2,2); % pan(2), tilt(2) の順つまり－48 度、チルトを－45 度にする。
を実行すると、
    nphControlCamera?Direction=PanRight
    nphControlCamera?Direction=TiltUp
と前回の位置をベースに、差分したコマンドを表示するようにするにはどのようにすればよいか？

ヒント：関数内部で現在位置を保持しておく必要がある。MATLAB 関数 function と persistent 変数を使う。
http://www.ikko.k.hosei.ac.jp/~matlab/faqtip.html
MATLAB 関数内のみで内部的に変数値を保持したい場合には？
（C 言語で static 変数に相当する機能を実現するには？）
を参考にするとよい。

MATLAB-Java を使った画像データの取り込み

CGI 仕様説明書によると、画像取り込みのためのコマンドは、
nphMotionJPEG
SnapshotJPEG
の 2 種類ある。SnapshotJPEG は、１枚のみの送信であり、携帯電話への画像の取り込みなどに使われる。MotionJPEG は、JPEG ファイルが連続的に送られてくるコマンドであり、Web ブラウザで動画などを見るときに使われるコマンドである。SnapshotJPEG は、Jpeg 画像データが送られてくるだけなので比較的容易に解析が可能である。それに対し、MotionJPEG は、連続的に Jpeg 画像データが送られてくるため取り扱いは若干厄介となる。そこでここでは、まず、SnapshotJPEG による例を示す。

SnapshotJPEG を使った例
MATLAB-Java により作成した、mclient3.m を以下に示す。

```
PORT = 80;
HOST=”192.168.1.3;
PASS=' MDEyMzQ6NTY3ODkw';
aSocket=java.net.Socket(HOST,PORT);
autoflush = 1;
out     = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
aReader = getInputStream(aSocket);
out.println('GET /SnapshotJPEG?Resolution=320x240&Quality=Standard HTTP/1.0');
aaa=['Authorization: Basic ',PASS];
out.println(aaa);
out.println('Host: localhost:8080');
out.println('Connection: Close');
out.println('');
loop = 1;
aaa=[];
bbb=[];
while (loop)
    s = read(aReader);
    aaa=[aaa char(s)];
    bbb=[bbb char(s)];
    index=strfind(aaa,'Content-length:');
    if ~isempty(index) break;end
end
aaa=[];
while (loop)
    s = read(aReader);
    aaa=[aaa char(s)];
    bbb=[bbb char(s)];
    index=strfind(aaa,char(10));
    if ~isempty(index) break;end
end
datanum=str2num(aaa);
aaa=[];
while (loop)
    s = read(aReader);
    bbb=[bbb char(s)];
    aaa=[aaa char(s)];
    index=strfind(aaa,['Content-type: image/jpeg',13,10,13,10]);
```

```
    if ~isempty(index) break;end
end
aaa=uint8(ones(1,datanum));
i=1;
while(loop)
    s = read(aReader);
    bbb=[bbb char(s)];
    lens = length(s);
    aaa(i+(0:(lens-1)))=uint8(s);
    if i >= datanum;break;end
    i = i + lens;
end
close(aSocket);
fid = fopen('hoge.jpg','wb');
fwrite(fid,aaa);
fclose(fid);
ccc=imread('hoge.jpg');
imshow(ccc);
```

ちなみに、実行後の、Web サーバからの応答を全て記録している変数 bbb の内部は、

```
>> bbb(1:100)
ans =
HTTP/1.0 200 OK
Expires: 0
Pragma: no-cache
Content-length: 16937
Content-type: image/jpeg


```

となっている。最後の□以降は、画像のバイナリコードである。このバイナリコードを JPEG ファイルとして保存し、imread 関数により読み込めば、画像データとして読み込むことができる。データフォーマットを見ると、Content-length により JPEG ファイルのバイト数、Content-type により JPEG ファイルであることがわかる。このスクリプトでは、まず、Content-lenth を strfind 関数により検出した後、そのバイト数を datanum に代入、次に、Content-type を strfind 関数により検出した後、それ以降のデータを JPEG ファイルとして datanum バイト分取り出している。また取り出したデータは、JPEG 形式で画像データが圧縮されて保存されているため、imread 関数によりファイルを介し、データ変換し表示している。
このプログラムの前後に tic,toc をつけると、その関数がかかった時間がわかる。
つまり、

```
>> tic;mclient2;toc
elapsed_time =
      3.6450
```

datanum は、圧縮の度合いにより変化するため、毎回異なる値になる。経過時間を見ると、1 枚の取り込み、表示にかかる時間は、約 3.6 秒かかるということである。

MATLAB-Java プログラムの最適化

Java プログラミングでの処理速度を高速化する手段として有効なものに、Buffered クラスの使用、MATLAB では、動的配列の確保をせずにあらかじめ配列の要素を決めて代入する方法がよく知られている。以下にそのような改善を行ったプログラム例を示す。

```
import java.io.*;
import java.net.*;
import java.util.*;
PORT = 80;
HOST="192.168.1.3;
PASS=' MDEyMzQ6NTY3ODkw';
aSocket=java.net.Socket(HOST,PORT);
```

```
autoflush = 1;
out     = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
aReader = java.io.BufferedInputStream(getInputStream(aSocket));
%aReader = getInputStream(aSocket);
out.println('GET /SnapshotJPEG?Resolution=320x240&Quality=Standard HTTP/1.0');
aaa=['Authorization: Basic ',PASS];
out.println(aaa);
out.println('Host: localhost:8080');
out.println('Connection: Close');
out.println('');
loop = 1;
aaa = aReader.read;
while (loop)
    avail=aReader.available;
    data=zeros(1,avail);
    for ii=1:avail
        data(ii)= aReader.read;
    end
    aaa=[aaa data];
    index=strfind(aaa,'Content-length:');
        if ~isempty(index) break;end
end
lens = length('Content-length:');
aaa=aaa(index(1)+lens:end);
while (loop)
    avail=aReader.available;
    data=zeros(1,avail);
    for ii=1:avail
        data(ii)= aReader.read;
    end
    aaa=[aaa data];
    index=strfind(aaa,10);
        if ~isempty(index(1)) break;end
end
datanum=str2num(char(aaa(1:index)));
aaa=aaa((index+1):end);
while (loop)
    avail=aReader.available;
    data=zeros(1,avail);
    for ii=1:avail
        data(ii)= aReader.read;
    end
    aaa=[aaa data];
        index=strfind(aaa,['Content-type: image/jpeg',13,10,13,10]);
        if ~isempty(index(1)) break;end
end
lens=length(['Content-type: image/jpeg',13,10,13,10]);
aaa=aaa((index+lens):end);
while (loop)
    avail=aReader.available;
    data=zeros(1,avail);
    for ii=1:avail
        data(ii)= aReader.read;
    end
    aaa=[aaa data];
    if(length(aaa)>=datanum)
        break;
    end
end
close(aSocket);
fid = fopen('hoge.jpg','wb');
fwrite(fid,aaa);
fclose(fid);
bbb=imread('hoge.jpg');
imshow(bbb);
```

このようにすることで若干であるが取り込み速度をアップすることができる。

JPEG 変換ライブラリによる JPEG からイメージデータの変換

処理速度の高速化を考えるとき、今まで実行してきたプログラムでは、JPEG イメージをファイルとして保存、その後、imread 関数によりイメージデータに変換している。この方法では、メモリより数十倍遅いハードディスクにデータを読み書きする必要があるため、処理速度を高速化したいとき、ボトルネックとなってしまう。そこで、MATLAB のバイナリ文字データを直接 RGB イメージデータに変換する方法が必要となる。では、どのようにして JPEG バイナリデータから img データに変換すればよいのであろうか？
MATLAB にある、imread 関数を解析してみて、その関数を文字列でも読み込み可能な形式に変換するのがもっとも効率的そうである。そこで、jpeg ファイルを解析している関数 M ファイルを探してみる。関数 M ファイルの中を見ていくと、最終的には、
C:¥MATLAB6p5¥toolbox¥matlab¥iofun¥private¥rjpgc.c
が受け持っていることがわかる。この中のコメントを見ると、次の記述がある。

```
This is a mex interface to the Independent Jpeg Group's (IJG)
LIBJPEG library.  This can read RGB and grayscale JPEG images.
The IJG code is available at:
ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6.tar.gz
```

つまり、MATLAB の JPG の画像変換は、IJG のライブラリを使っている。現状でのプログラムでは、ファイルからデータを直接読み込む形式になっているため、ライブラリの使い方を見直さないと改良が不可能である。そこでまず、現在の IJG の最新コードを
ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz
から FTP により取得する。このファイルの解凍は、Linux/Unix 上であるならば、tar で、Windows 上ならば、例えば、Lhaplus などを使う。この IJG ライブラリは、Windows プラットフォームだけではなく、さまざまなプラットフォームに対応している。コンパイルの方法もそれぞれプラットフォームに合わせ行う必要がある。ここでは、Windows 版の手順を紹介する。これを実行する前に VisualC++６をインストールし、コマンドラインが使える環境にしておく必要がある。
まず、コマンドプロンプト上で

```
copy makefile.vc makefile
copy jconfig.vc jconfig.h
nmake -f makelib.ds
```

を実行する。実行すると、Release フォルダに jpeg.lib ができる。このファイルをひとつしたのフォルダにコピーし、

```
copy jpeg.lib ../jpeg.lib
ren jpeg.lib libjpeg.lib
```

として名前を変更する。名前が変更し終わったら、

```
nmake
```

と実行すれば一通りの検証用ツールなどの作成が可能となる。
ちなみに、MATLAB では、libjpeg.lib のあるフォルダに、rjpgc.c をコピーし、

```
>> mex rjpgc.c libjpeg.lib
```

を実行すると、rjpgc.dll
が作ることができる。これでコンパイルできる環境ができたわけであるが、このままでは、ファイルから Jpeg 形式のファイルを読み込み変換するだけである。
ここで行いたいのは、JPEG 文字列から画像 RGB イメージへの変換である。
IJG に付属しているライブラリドキュメントを注意深く読んでみると、
libjpeg.doc

```
Compressed data handling (source and destination managers)
----------------------------------------------------------
The JPEG compression library sends its compressed data to a "destination
manager" module.  The default destination manager just writes the data to a
```

stdio stream, but you can provide your own manager to do something else.
Similarly, the decompression library calls a "source manager" to obtain the compressed data; you can provide your own source manager if you want the data to come from somewhere other than a stdio stream.

このドキュメントの大半は、一般的な使用例であるファイルを介した JPEG データの変換のやり方が書いてあるのであるが、ここを読むと source manager モジュールにより stdio ストリーム（ファイルからのデータ）以外にも設定可能であるとのことが書いてある。つまり、source manager モジュールをファイル用ではなく文字列用に変更すればよい。
また、もうちょっと読んでみると、

A data source manager provides five methods:
init_source (j_decompress_ptr cinfo)
Initialize source. This is called by jpeg_read_header() before any data is actually read. Unlike init_destination(), it may leave bytes_in_buffer set to 0 (in which case a fill_input_buffer() call will occur immediately).

fill_input_buffer (j_decompress_ptr cinfo)
This is called whenever bytes_in_buffer has reached zero and more data is wanted. In typical applications, it should read fresh data into the buffer (ignoring the current state of next_input_byte and bytes_in_buffer), reset the pointer & count to the start of the buffer, and return TRUE indicating that the buffer has been reloaded. It is not necessary to fill the buffer entirely, only to obtain at least one more byte. bytes_in_buffer MUST be set to a positive value if TRUE is returned. A FALSE return should only be used when I/O suspension is desired (this mode is discussed in the next section).

skip_input_data (j_decompress_ptr cinfo, long num_bytes)
Skip num_bytes worth of data. The buffer pointer and count should be advanced over num_bytes input bytes, refilling the buffer as needed. This is used to skip over a potentially large amount of uninteresting data (such as an APPn marker). In some applications it may be possible to optimize away the reading of the skipped data, but it's not clear that being smart is worth much trouble; large skips are uncommon. bytes_in_buffer may be zero on return. A zero or negative skip count should be treated as a no-op.

resync_to_restart (j_decompress_ptr cinfo, int desired)
This routine is called only when the decompressor has failed to find a restart (RSTn) marker where one is expected. Its mission is to find a suitable point for resuming decompression. For most applications, we recommend that you just use the default resync procedure, jpeg_resync_to_restart(). However, if you are able to back up in the input data stream, or if you have a-priori knowledge about the likely location of restart markers, you may be able to do better. Read the read_restart_marker() and jpeg_resync_to_restart() routines in jdmarker.c if you think you'd like to implement your own resync procedure.

term_source (j_decompress_ptr cinfo)
Terminate source --- called by jpeg_finish_decompress() after all data has been read. Often a no-op.

source manager は、5 つの関数によりデータの入力を管理している。source manager は jdatasrc.c の中に定義されている。これらの関数の一部を置き換えれば、ファイルからでなく、文字列から RGB イメージデータに変換できる。MATLAB で用意されている rjpgc.c のプログラムは、この jdatasrc.c の中の jpeg_stdio_src 関数を呼び出している。つまり、この関数をファイル用から文字列用に変更する。そこで jdatasrc.c での実装を参考に、関連のある４の関数について文字列用に修正した。既存の rjpgc.c と区別するために、jpgstr2img.c とリネームしたものとして解説を進める。
ここでの変更点は、jdatasrc.c では、ファイル操作のためバッファを設け、逐次的にデータを取り

込んでいたのに対し、文字列で扱う場合には、一度に一気に読み込めるため、バッファを設けず、読み込んだ結果を全部、JPEG ライブラリ関数に渡すように変更している。

```
typedef struct {
    struct jpeg_source_mgr pub; /* public fields */
} my_source_mgr;

typedef my_source_mgr * my_src_ptr;

METHODDEF(void) init_source (j_decompress_ptr cinfo) {
    my_src_ptr src = (my_src_ptr) cinfo->src;
}
METHODDEF(boolean) fill_input_buffer (j_decompress_ptr cinfo) {
    my_src_ptr src = (my_src_ptr) cinfo->src;
    src->pub.next_input_byte = infile;
    src->pub.bytes_in_buffer = maxlen;
    return TRUE;
}
METHODDEF(void) skip_input_data (j_decompress_ptr cinfo, long num_bytes) {
    my_src_ptr src = (my_src_ptr) cinfo->src;
    if (num_bytes > 0) {
        while (num_bytes > (long) src->pub.bytes_in_buffer) {
            num_bytes -= (long) src->pub.bytes_in_buffer;
            (void) fill_input_buffer(cinfo);
        }
        src->pub.next_input_byte += (size_t) num_bytes;
        src->pub.bytes_in_buffer -= (size_t) num_bytes;
    }
}

METHODDEF(void) term_source (j_decompress_ptr cinfo){}

jpeg_string_src(j_decompress_ptr cinfo) {
    my_src_ptr src;
    if (cinfo->src == NULL) { /* first time for this JPEG object? */
        cinfo->src = (struct jpeg_source_mgr *)(*cinfo->mem->alloc_small) ((j_common_ptr) cinfo,
JPOOL_PERMANENT,sizeof(my_source_mgr));
        src = (my_src_ptr) cinfo->src;
    }
    src = (my_src_ptr) cinfo->src;
    src->pub.init_source = init_source;
    src->pub.fill_input_buffer = fill_input_buffer;
    src->pub.skip_input_data = skip_input_data;
    src->pub.resync_to_restart = jpeg_resync_to_restart; /* use default method */
    src->pub.term_source = term_source;
    src->pub.bytes_in_buffer = 0; /* forces fill_input_buffer on first read */
    src->pub.next_input_byte = NULL; /* until buffer loaded */
}
```

これら関数を MEX 関数内に入れ、
後は、rjpgc.c の中のファイルからデータを取り込む関数

```
jpeg_stdio_src(&cinfo, infile);
```

を

```
jpeg_string_src(&cinfo);
```

に変更、また、今までファイル名を読み込んでいたところを、長い文字列を受け取れるように変更、そのための mxFree の場所の変更し、
mex strjpg2img.c
としてコンパイルすればよい。
以下に jpgstr2img.c を示す。なお、rjpgc.c のコンパイルには、libjpeg.lib のライブラリを指定する必要があったが、この jpgstr2img.c では、その必要はない。理由は、プログラム内部の#pragma でライブラリを指定しているためである。

```
#include "mex.h"
#include <stdio.h>
#include <string.h>
```

```
#include <setjmp.h>
#include "jpeglib.h"
#pragma comment(lib, "libjpeg.lib")

static mxArray *ReadRgbJPEG(j_decompress_ptr cinfoPtr);
static mxArray *ReadGrayJPEG(j_decompress_ptr cinfoPtr);
static void my_error_exit (j_common_ptr cinfo);
static void my_output_message (j_common_ptr cinfo);
static unsigned char * infile;      /* source stream */
static long maxlen = 0;

struct my_error_mgr {
    struct jpeg_error_mgr pub;   /* "public" fields */
    jmp_buf setjmp_buffer;    /* for return to caller */
};

typedef struct my_error_mgr *my_error_ptr;

typedef struct {
    struct jpeg_source_mgr pub; /* public fields */
} my_source_mgr;

typedef my_source_mgr * my_src_ptr;

METHODDEF(void) init_source (j_decompress_ptr cinfo) {
    my_src_ptr src = (my_src_ptr) cinfo->src;
}
METHODDEF(boolean) fill_input_buffer (j_decompress_ptr cinfo) {
    my_src_ptr src = (my_src_ptr) cinfo->src;
    src->pub.next_input_byte = infile;
    src->pub.bytes_in_buffer = maxlen;
    return TRUE;
}
METHODDEF(void) skip_input_data (j_decompress_ptr cinfo, long num_bytes) {
    my_src_ptr src = (my_src_ptr) cinfo->src;
    if (num_bytes > 0) {
        while (num_bytes > (long) src->pub.bytes_in_buffer) {
            num_bytes -= (long) src->pub.bytes_in_buffer;
            (void) fill_input_buffer(cinfo);
        }
        src->pub.next_input_byte += (size_t) num_bytes;
        src->pub.bytes_in_buffer -= (size_t) num_bytes;
    }
}

METHODDEF(void) term_source (j_decompress_ptr cinfo){}

jpeg_string_src(j_decompress_ptr cinfo) {
    my_src_ptr src;
    if (cinfo->src == NULL) { /* first time for this JPEG object? */
        cinfo->src = (struct jpeg_source_mgr *)(*cinfo->mem->alloc_small) ((j_common_ptr) cinfo,
JPOOL_PERMANENT,sizeof(my_source_mgr));
        src = (my_src_ptr) cinfo->src;
    }
    src = (my_src_ptr) cinfo->src;
    src->pub.init_source = init_source;
    src->pub.fill_input_buffer = fill_input_buffer;
    src->pub.skip_input_data = skip_input_data;
    src->pub.resync_to_restart = jpeg_resync_to_restart; /* use default method */
    src->pub.term_source = term_source;
    src->pub.bytes_in_buffer = 0; /* forces fill_input_buffer on first read */
    src->pub.next_input_byte = NULL; /* until buffer loaded */
}

void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[]) {
    mxArray *outArray;
    int32_T strlen;
    int i;
```

```
    struct jpeg_decompress_struct cinfo;
    struct my_error_mgr jerr;
    int current_row;
    if (nrhs < 1) mexErrMsgTxt("Not enough input arguments.");
    if(! mxIsUint8(prhs[0])) mexErrMsgTxt("First argument is not a uint8.");
    maxlen = mxGetM(prhs[0]) * mxGetN(prhs[0]) + 1;
    infile = (unsigned char*)mxCalloc(maxlen,sizeof(char));
    memcpy(infile,(unsigned char *)mxGetPr(prhs[0]),maxlen);
    cinfo.err = jpeg_std_error(&jerr.pub);
    jerr.pub.output_message = my_output_message;
    jerr.pub.error_exit = my_error_exit;
    if(setjmp(jerr.setjmp_buffer)) {
        jpeg_destroy_decompress(&cinfo);
        mxFree(infile);
        return;
    }
    jpeg_create_decompress(&cinfo);
    jpeg_string_src(&cinfo);
    jpeg_read_header(&cinfo, TRUE);
    jpeg_start_decompress(&cinfo);
    if (cinfo.output_components == 1) outArray = ReadGrayJPEG(&cinfo);
    else outArray = ReadRgbJPEG(&cinfo);
    jpeg_finish_decompress(&cinfo);
    jpeg_destroy_decompress(&cinfo);
    plhs[0]=outArray;
    return;
}
static mxArray * ReadRgbJPEG(j_decompress_ptr cinfoPtr) {
    long i,j,k,row_stride;
    int dims[3];                    /* For the call to mxCreateNumericArray */
    mxArray *img;
    JSAMPARRAY buffer;
    int current_row;
    uint8_T *pr_red, *pr_green, *pr_blue;
    row_stride = cinfoPtr->output_width * cinfoPtr->output_components;
    buffer = (*cinfoPtr->mem->alloc_sarray)((j_common_ptr) cinfoPtr, JPOOL_IMAGE, row_stride, 1);
    dims[0]   = cinfoPtr->output_height;
    dims[1]   = cinfoPtr->output_width;
    dims[2]   = 3;
    img = mxCreateNumericArray(3, dims, mxUINT8_CLASS, mxREAL);
    pr_red    = (uint8_T *) mxGetData(img);
    pr_green = pr_red + (dims[0]*dims[1]);
    pr_blue   = pr_red + (2*dims[0]*dims[1]);
    while (cinfoPtr->output_scanline < cinfoPtr->output_height) {
        current_row = cinfoPtr->output_scanline; /* Temp var won't get ++'d */
        jpeg_read_scanlines(cinfoPtr, buffer,1); /*   by jpeg_read_scanlines */
        for (i=0;i<cinfoPtr->output_width;i++) {
            j=(i)*cinfoPtr->output_height+current_row;
            pr_red[j]     = buffer[0][i*3+0];
            pr_green[j] = buffer[0][i*3+1];
            pr_blue[j]   = buffer[0][i*3+2];
        }
    }
    return img;
}
static mxArray * ReadGrayJPEG(j_decompress_ptr cinfoPtr) {
    long i,j,k,row_stride;
    int dims[3];                    /* For the call to mxCreateNumericArray */
    mxArray *img;
    JSAMPARRAY buffer;
    int current_row;
    uint8_T *pr_gray;
    row_stride = cinfoPtr->output_width * cinfoPtr->output_components;
    buffer = (*cinfoPtr->mem->alloc_sarray)((j_common_ptr) cinfoPtr, JPOOL_IMAGE, row_stride, 1);
    dims[0]   = cinfoPtr->output_height;
    dims[1]   = cinfoPtr->output_width;
    dims[2]   = 1;
    img = mxCreateNumericArray(2, dims, mxUINT8_CLASS, mxREAL);
```

```
    pr_gray     = (uint8_T *) mxGetData(img);
    while (cinfoPtr->output_scanline < cinfoPtr->output_height) {
        current_row=cinfoPtr->output_scanline; /* Temp var won't get ++'d */
        jpeg_read_scanlines(cinfoPtr, buffer,1); /*    by jpeg_read_scanlines */
        for (i=0;i<cinfoPtr->output_width;i++) {
            j=(i)*cinfoPtr->output_height+current_row;
            pr_gray[j]     = buffer[0][i];
        }
    }
    return img;
}
static void my_error_exit (j_common_ptr cinfo) {
    my_error_ptr myerr = (my_error_ptr) cinfo->err;
    (*cinfo->err->output_message) (cinfo);
    longjmp(myerr->setjmp_buffer, 1);
}
static void my_output_message (j_common_ptr cinfo) {
    char buffer[JMSG_LENGTH_MAX];
    (*cinfo->err->format_message) (cinfo, buffer);
    mexWarnMsgTxt(buffer);
}
```

これで完成である。この関数は、JPG 文字列から直接イメージデータに変換できる。
以下のコードを使えば動作をチェックすることができる。

```
fid = fopen('hogehoge.jpg');
aaa=uint8(fread(fid));
fclose(fid);
ccc=jpgstr2img(aaa);
imshow(ccc);
```

つまり、今まで、

```
fid = fopen('hoge.jpg','wb');
fwrite(fid,aaa);
fclose(fid);
ccc=imread('hoge.jpg');
imshow(ccc);
```

としていたコードが、

```
ccc=jpgstr2img(aaa);
imshow(ccc);
```

となり、一度、ファイルに保存してから、もう一度読み込む手間を省くことが可能になる。

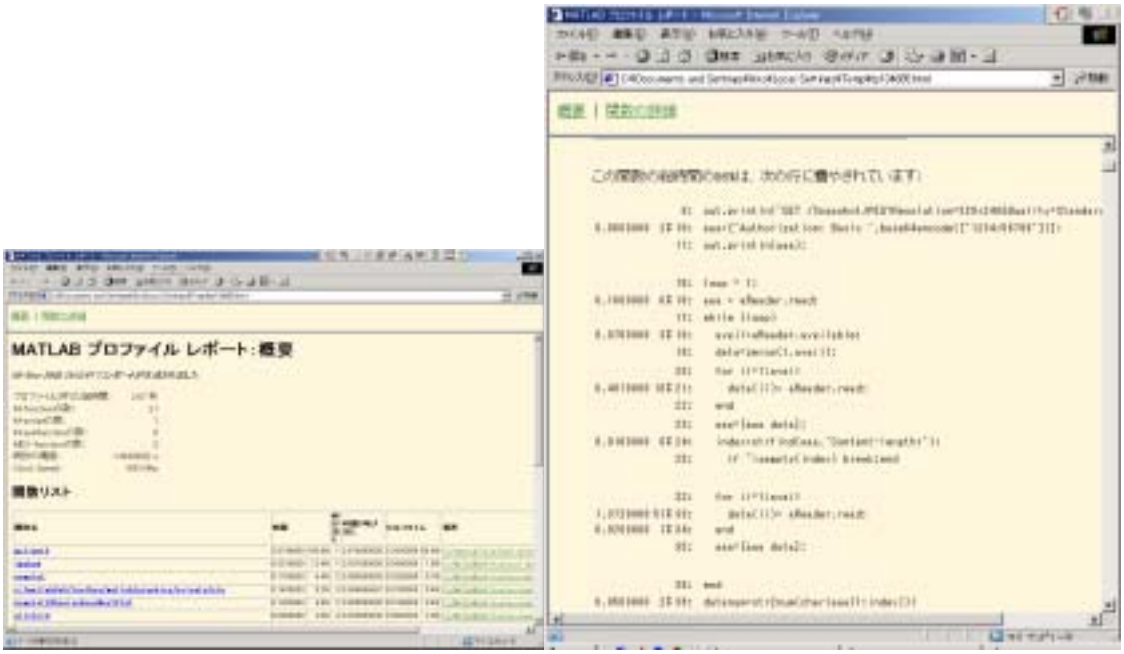
MATLAB の profile 関数による処理速度の評価

mclient2.m

```
import java.io.*;
import java.net.*;
import java.util.*;
PORT = 80;
HOST="192.168.1.3;
PASS=' MDEyMzQ6NTY3ODkw';
aSocket=java.net.Socket(HOST,PORT);
autoflush = 1;
out     = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
aReader = getInputStream(aSocket);
out.println('GET /SnapshotJPEG?Resolution=320x240&Quality=Standard HTTP/1.0');
aaa=['Authorization: Basic ',PASS];
out.println(aaa);
out.println('Host: localhost:8080');
out.println('Connection: Close');
out.println('');
loop = 1;
```

```
aaa = aReader.read;
while (loop)
    avail=aReader.available;
    data=zeros(1,avail);
    for ii=1:avail
        data(ii)= aReader.read;
    end
    aaa=[aaa data];
    index=strfind(aaa,'Content-length:');
        if ~isempty(index) break;end
end
lens = length('Content-length:');
aaa=aaa(index(1)+lens:end);
while (loop)
    avail=aReader.available;
    data=zeros(1,avail);
    for ii=1:avail
        data(ii)= aReader.read;
    end
    aaa=[aaa data];
    index=strfind(aaa,10);
        if ~isempty(index(1)) break;end
end
datanum=str2num(char(aaa(1:index)))
aaa=aaa((index+1):end);
while (loop)
    avail=aReader.available;
    data=zeros(1,avail);
    for ii=1:avail
        data(ii)= aReader.read;
    end
    aaa=[aaa data];
        index=strfind(aaa,['Content-type: image/jpeg',13,10,13,10]);
        if ~isempty(index(1)) break;end
end
lens=length(['Content-type: image/jpeg',13,10,13,10]);
aaa=aaa((index+lens):end);
while (loop)
    avail=aReader.available;
    data=zeros(1,avail);
    for ii=1:avail
        data(ii)= aReader.read;
    end
    aaa=[aaa data];
    if(length(aaa)>=datanum)
        break;
    end
end
close(aSocket);
imshow(jpgstr2img(uint8(aaa)));
```

profile on; mclient2;profile report
を実行すると、どの関数がどの程度の時間がかかっているかわかる。



ここで、mclient2 の中を見てみると、ネットワークからの read 命令が特に時間を費やしていることがわかる。このように profile 関数を使うことで、MATLAB アプリケーションでどの部分が時間を消費しているのかが容易に把握することができる。

Winsock2 によるデータ取り込み関数の作成

Profile 関数により、読み込み速度がボトルネックになっていることがわかった。そこでここでは、大量にデータを読み込む画像データ取り込みに絞り、専用の MEX 関数を作成する。取り込み速度の高速化のため、ネットワークのプログラミングでは Winsock2 の非同期通信 API を活用する。以下に、C++バージョンのネットワークからのデータ取り込みプログラムを示す。

```
// mex mhttp2jpgstr.cpp
#include "mex.h"
#include <string.h>
#include <ws2spi.h>
#pragma comment(lib, "ws2_32.lib")

#define BUFMAX 65535

unsigned long gethostaddress(LPSTR szBuff) {
    PHOSTENT phe;
    struct in_addr in;
    phe = gethostbyname(szBuff);
    if(phe == NULL) exit(1);
    memcpy(&in,phe->h_addr,sizeof(in));
//  mexPrintf("Host name : %s¥n",phe->h_name);
//  mexPrintf("IP address: %s¥n",inet_ntoa(in));
    return *((unsigned long *)phe->h_addr);
}
int Recv(SOCKET s, char *inbuf, int len, int flags) {
    int rc = 0;
    int rc1;
    fd_set rset;
    struct timeval rtmout;
    for(;;) {
        rc = recv(s, inbuf, len, flags);
        if(rc > 0) return rc;
        if(WSAGetLastError() == WSAEWOULDBLOCK) {
            FD_ZERO(&rset);
```

```
            FD_SET(s, &rset);
            rtmout.tv_sec = 5;
            rtmout.tv_usec = 0;
            rc1 = select(FD_SETSIZE, &rset, NULL, NULL, &rtmout);
            if(rc1 == 0) {
                mexPrintf("select RECV TIMEOUT:¥n");
                return -1;
            }
            if(rc1 == SOCKET_ERROR) {
                mexPrintf("select RECV ERROR: err=%d¥n", WSAGetLastError());
                return -1;
            }
            if(!FD_ISSET(rset.fd_array[0], &rset)) {
                mexPrintf("Select RECV wakes but FD_ISSET() FAILS¥n");
                return -1;
            }
            continue;
        }
        return -1;
    }
}
static SOCKET Socket = NULL;
static int start = 0;
void closeall(void) {
    WSACleanup();
    start = 0;
    mexPrintf("¥nClear mhttp2jpgstr by Gerox(c) 2003¥n");
}
static WORD wVerReq;
static WSADATA wsadata;
void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[]) {
    if(nrhs != 3) {
        mexPrintf(" Image captureing tool for Panasonic Network camera¥n");
        mexPrintf(" KX-HCM1, KX-HCM2,KX-HCM130,KX-HCM170,KX-HCM180¥n");
        mexPrintf("Usage: str = mhttp2jpgstr(hostname,portnum,encryptedpass)¥n");
        mexPrintf("Example: str = mhttp2jpgstr('192.168.1.3',80,'MTIzNDo1Njc4OQ==');¥n");
        mexPrintf("-----------------------------------------------¥n");
        mexPrintf("%s    by Gerox(c) Build %s¥n",__FILE__,__DATE__);
        mexPrintf("-----------------------------------------------¥n");
        return;
    }
    int hostnamelen = mxGetM(prhs[0])*mxGetN(prhs[0])+1;
    char hostname[255];
    mxGetString(prhs[0],hostname,hostnamelen);
    int portnum = (int) mxGetScalar(prhs[1]);
    char password[255];
    int passwordlen = mxGetM(prhs[2])*mxGetN(prhs[2])+1;
    mxGetString(prhs[2],password,passwordlen);
    if(start == 0) {
        wVerReq = MAKEWORD(2,0);
        if(WSAStartup(wVerReq,&wsadata) != 0) return;
        mexAtExit(closeall);
        start = 1;
    }
    Socket = WSASocket(AF_INET,SOCK_STREAM,0,NULL,0,WSA_FLAG_OVERLAPPED);
    if(Socket == INVALID_SOCKET) {
        mexPrintf("socket FAILED: err=%d¥n", WSAGetLastError());
        return;
    }

    char recMsg[BUFMAX],sendMsg[BUFMAX];
    char * curMsg;
    int len,i;
    struct sockaddr_in clientName;
    memset(&clientName,0x00,sizeof(clientName));
    clientName.sin_family = AF_INET;
    clientName.sin_addr.s_addr = gethostaddress(hostname);
    clientName.sin_port = htons((unsigned short)portnum);
```

```
    Sleep(0);
    if(connect(Socket,(struct sockaddr FAR *)&clientName,sizeof(clientName)) < 0) {
        mexPrintf("Connection fail %d¥n", WSAGetLastError());
        return;
    }
    int number = sizeof(number);
    if(setsockopt(Socket,getprotobyname("tcp")->p_proto,TCP_NODELAY,(const char *)&number,sizeof(number))) {
        mexPrintf("setsockopt FAILED: err=%d¥n", WSAGetLastError());
        return;
    }
    unsigned long ul = 1;
    if(ioctlsocket(Socket, FIONBIO, &ul )) {
        mexPrintf("ioctlsocket FIONBIO Child: FAILED err=%d¥n",WSAGetLastError());
        return;
    }
    int recvbufsz = BUFMAX;
    if(setsockopt(Socket,SOL_SOCKET,SO_RCVBUF,(char *)&recvbufsz,sizeof(recvbufsz))) {
        mexPrintf("setsockopt(SO_RCVBUF %d FAILED: err=%d¥n",recvbufsz, WSAGetLastError());
        return;
    }
    sprintf(sendMsg,"GET /SnapshotJPEG?Resolution=320x240&Quality=Standard HTTP/1.0¥n¥r");
    if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
    sprintf(sendMsg,"Authorization: Basic %s¥n¥r",password);
    if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
    sprintf(sendMsg,"Connection: Close¥n¥r");
    if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
    sprintf(sendMsg,"¥n¥r¥n¥r");
    if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
    int sendbufsz = BUFMAX;
    if(setsockopt(Socket,SOL_SOCKET,SO_SNDBUF,(char *)&sendbufsz,sizeof(sendbufsz))) {
        mexPrintf("setsockopt(SO_SNDBUF %d FAILED: err=%d¥n",sendbufsz, WSAGetLastError());
        return;
    }
    int totallen = 0;
    curMsg = (char *)recMsg;
    while(1) {
        Sleep(0);
        len = Recv(Socket,curMsg,BUFMAX,0);
        if(len == -1) break;
        totallen += len;
        curMsg += len;
    }
    closesocket(Socket);
    curMsg = (char *)recMsg;
    len = totallen;
    for(i = 0;i < 120;i++) {
        if((unsigned char)curMsg[i] == 0xff) {
            if((unsigned char)curMsg[i + 1] == 0xd8) {
                totallen = len - i;
// memmove maybe better memcpy for speed
                memcpy((unsigned char *)recMsg,(unsigned char *)(curMsg + i),totallen);
                curMsg = (char *)recMsg;
                break;
            }
        }
    }
/*
    FILE *fp = fopen("hogehoge.jpg","wb");
    fwrite(curMsg,sizeof(char),totallen,fp);
    fclose(fp);
*/
    int dims[2];
    dims[0]   = totallen;
    dims[1]   = 1;
    plhs[0] = mxCreateNumericArray(2, dims, mxUINT8_CLASS, mxREAL);
    char * matptr = (char *)mxGetPr(plhs[0]);
    memcpy((char *)matptr,(char *)curMsg,totallen);
    return;
```

```
}
```

この関数の使い方は、常に３つの引数をとり、

```
PORT = 80;
HOST=”192.168.1.3;
PASS=’ MDEyMzQ6NTY3ODkw’;
jpgstr=mhttp2jpgstr(HOST,PORT,PASS);
imshow(jpgstr2img(jpgstr));
```

このような使い方をする。2 番目はポート番号をあらわす。一般に公開されているネットワークカメラでは、80 番ポートを使う場合が多い。また、パスワードが無い場合でも第三引数がないと動かないので適当な文字を設定しておく必要がある。
以上のプログラムを用い、プログラミングした例を samplejpg.m を以下に示す。

```
PORT = 80;
HOST=”192.168.1.3;
PASS=’ MDEyMzQ6NTY3ODkw’;
figure(1);set(1,'doublebuffer','on','Interruptible','on');
jpgstr=mhttp2jpgstr(HOST,PORT,PASS);
abc=imshow(jpgstr2img(jpgstr));
try
    for i=1:100
        tic
        jpgstr = mhttp2jpgstr(HOST,PORT,PASS);
        toc
        imshow(jpgstr2img(jpgstr));
        drawnow;
    end
catch
    clear mex
end
clear mex
```

このプログラムを実行すると、一枚の画像を取り込む時間は、約 0.7 秒となった。
また、このプログラムに、以前作成した、カメラ制御用スクリプトを融合し、画像上をクリックすることで画面が変化するようにプログラムすることができる。
そのためには、２つのプログラムが必要となる。

samplejpg.m

```
import java.io.*;
import java.net.*;
import java.util.*;
PORT = 80;
HOST=”192.168.1.3;
PASS=’ MDEyMzQ6NTY3ODkw’;
figure(1);set(1,'doublebuffer','on','Interruptible','on');
jpgstr=mhttp2jpgstr(HOST,PORT,PASS);
abc=imshow(jpgstr2img(jpgstr));
set(abc,'ButtonDownFcn','mclient4;');
try
    for i=1:100
        tic;
        jpgstr = mhttp2jpgstr(HOST,PORT,PASS);
        toc
        set(abc,'CData',jpgstr2img(jpgstr));
        drawnow;
    end
catch
    clear mex;
end
clear mex
```

mclient4.m

```
pos=get(gca,'CurrentPoint');
```

```
pos=pos(1,1:2);
psize=get(gca,'PlotBoxAspectRatio');
psize=psize(1:2);
direction=(psize/2-pos);
if direction(1)>0
    motion1 = 'PanLeft';
else
    motion1 = 'PanRight';
end
if direction(2)>0
    motion2 = 'TiltUp';
else
    motion2 = 'TiltDown';
end
aSocket=java.net.Socket(HOST,PORT);
autoflush = 1;
out     = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
aaa=['GET /nphControlCamera?Direction=',motion1,' HTTP/1.0'];
out.println(aaa);
aaa=['Authorization: Basic ',PASS];
out.println(aaa);
out.println('Host: localhost:8080');
out.println('Connection: Close');
out.println('');
close(aSocket);
aSocket=java.net.Socket(HOST,PORT);
out     = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
aaa=['GET /nphControlCamera?Direction=',motion2,' HTTP/1.0'];
out.println(aaa);
aaa=['Authorization: Basic ',PASS];
out.println(aaa);
out.println('Host: localhost:8080');
out.println('Connection: Close');
out.println('');
close(aSocket);
```

MotionJPEG 機能による連続データ取り込み

CGI の説明書によれば、MotionJPEG による連続データ取り込みモードがある。
このモードを使うと、SnapShot モードのように毎回、画像データの要求を出さなくても、連続的に JPEG データを送信するモードとなる。
そこで、C++のプログラムをその仕様に変更した。JPEG では、一回に送られる画像のバイト数は、圧縮の程度により変化するため予測できない。そこで JPEG データの始まりと終わりのチェックには、16 進コードで、FFD8、FFD9 を検出し、検出ができたら 1 枚のデータとして出力するように設計している。また、このプログラムが動かされない場合、データがバッファに溜まってしまい、リアルタイム性がなくなってしまうので、2 回目以降画像取り込みが行われるときは、なるべく最新の画像データを取り込むように設計している。以下にその C++プログラムのソースコードを示す。

mhttp2mpgstr.cpp

```
// mex mhttp2mpgstr.cpp
#include "mex.h"
#include <string.h>
#include <ws2spi.h>
#pragma comment(lib, "ws2_32.lib")

#define BUFMAX 65535
static char sendMsg[BUFMAX];
char *recMsg;
char *curMsg;

static unsigned long gethostaddress(LPSTR szBuff) {
    PHOSTENT phe;
    struct in_addr in;
```

```
    phe = gethostbyname(szBuff);
    if(phe == NULL) exit(1);
    memcpy(&in,phe->h_addr,sizeof(in));
    mexPrintf("Host name : %s¥n",phe->h_name);
    mexPrintf("IP address: %s¥n",inet_ntoa(in));
    return *((unsigned long *)phe->h_addr);
}
int Recv(SOCKET s, char *inbuf, int len, int flags) {
    int rc = 0;
    int rc1;
    fd_set rset;
    struct timeval rtmout;
    for(;;) {
        rc = recv(s, inbuf, len, flags);
        if(rc > 0) return rc;
        if(WSAGetLastError() == WSAEWOULDBLOCK) {
            FD_ZERO(&rset);
            FD_SET(s, &rset);
            rtmout.tv_sec = 5;
            rtmout.tv_usec = 0;
            rc1 = select(FD_SETSIZE, &rset, NULL, NULL, &rtmout);
            if(rc1 == 0) {
                mexPrintf("select RECV TIMEOUT:¥n");
                return -1;
            }
            if(rc1 == SOCKET_ERROR) {
                mexPrintf("select RECV ERROR: err=%d¥n", WSAGetLastError());
                return -1;
            }
            if(!FD_ISSET(rset.fd_array[0], &rset)) {
                mexPrintf("Select RECV wakes but FD_ISSET() FAILS¥n");
                return -1;
            }
            continue;
        }
        return -1;
    }
}
static SOCKET Socket = NULL;
static int start = 0;
void closeall(void) {
    mxFree(recMsg);
    closesocket(Socket);
    WSACleanup();
    start = 0;
    mexPrintf("¥nClear mhttpmeg2str by Gerox(c) 2003¥n");
}
static struct sockaddr_in clientName;
static int number;
static unsigned long ul = 1;
static WORD wVerReq;
static WSADATA wsadata;
static int recvbufsz = BUFMAX;
static int sendbufsz = BUFMAX;
static int offset = 0;
void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[]) {
    int len = 0;
    int stat = 0;
    if(start == 0) {
        if(nrhs != 3) {
            mexPrintf(" Image captureing tool for Panasonic Network camera¥n");
            mexPrintf(" KX-HCM1, KX-HCM2,KX-HCM130,KX-HCM170,KX-HCM180¥n");
            mexPrintf("Usage: str = mhttp2mpgstr(hostname,portnum,encryptedpass)¥n");
            mexPrintf("Example: str = mhttp2mpgstr('www.ikko.k.hosei.ac.jp',80,'MTIzNDo1Njc4OQ==');¥n");
            mexPrintf("Example: str = mhttp2mpgstr('192.168.1.3',80,'MTIzNDo1Njc4OQ==');¥n");
            mexPrintf("-----------------------------------------¥n");
            mexPrintf("%s    by Gerox(c) Build %s¥n",__FILE__,__DATE__);
            mexPrintf("-----------------------------------------¥n");
```

```
        return;
    }
    wVerReq = MAKEWORD(2,0);
    if(WSAStartup(wVerReq,&wsadata) != 0) return;
    recMsg = (char *)mxCalloc(BUFMAX,sizeof(char));
    mexMakeMemoryPersistent(recMsg);
    mexAtExit(closeall);
    start = 1;
    int hostnamelen = mxGetM(prhs[0])*mxGetN(prhs[0])+1;
    char hostname[255];
    mxGetString(prhs[0],hostname,hostnamelen);
    int portnum = (int) mxGetScalar(prhs[1]);
    char password[255];
    int passwordlen = mxGetM(prhs[2])*mxGetN(prhs[2])+1;
    mxGetString(prhs[2],password,passwordlen);

    Socket = WSASocket(AF_INET,SOCK_STREAM,0,NULL,0,WSA_FLAG_OVERLAPPED);
    if(Socket == INVALID_SOCKET) {
        mexPrintf("socket FAILED: err=%d¥n", WSAGetLastError());
        return;
    }
    memset(&clientName,0x00,sizeof(clientName));
    clientName.sin_family = AF_INET;
    clientName.sin_addr.s_addr = gethostaddress(hostname);
    clientName.sin_port = htons((unsigned short)portnum);
    Sleep(0);
    if(connect(Socket,(struct sockaddr FAR *)&clientName,sizeof(clientName)) < 0) {
        mexPrintf("Connection fail %d¥n", WSAGetLastError());
        return;
    }
    number = sizeof(number);
    if(setsockopt(Socket,getprotobyname("tcp")->p_proto,TCP_NODELAY,(const char *)&number,sizeof(number))) {
        mexPrintf("setsockopt FAILED: err=%d¥n", WSAGetLastError());
        return;
    }
    ul = 1;
    if(ioctlsocket(Socket, FIONBIO, &ul )) {
        mexPrintf("ioctlsocket FIONBIO Child: FAILED err=%d¥n",WSAGetLastError());
        return;
    }
    recvbufsz = BUFMAX;
    if(setsockopt(Socket,SOL_SOCKET,SO_RCVBUF,(char *)&recvbufsz,sizeof(recvbufsz))) {
        mexPrintf("setsockopt(SO_RCVBUF %d FAILED: err=%d¥n",recvbufsz, WSAGetLastError());
        return;
    }
    sendbufsz = BUFMAX;
    if(setsockopt(Socket,SOL_SOCKET,SO_SNDBUF,(char *)&sendbufsz,sizeof(sendbufsz))) {
        mexPrintf("setsockopt(SO_SNDBUF %d FAILED: err=%d¥n",sendbufsz, WSAGetLastError());
        return;
    }
    sprintf(sendMsg,"GET /nphMotionJPEG?Resolution=320x240&Quality=Standard HTTP/1.0¥n¥r");
    if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
    sprintf(sendMsg,"Authorization: Basic %s¥n¥r",password);
    if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
    sprintf(sendMsg,"Connection: Close¥n¥r");
    if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
    sprintf(sendMsg,"¥n¥r¥n¥r");
    if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
    curMsg = recMsg;
    offset = 0;
} else {
    stat = 0;
    curMsg = recMsg + offset;
    len = Recv(Socket,curMsg,BUFMAX-offset,0);
    curMsg -= offset;
    len += offset;
    offset = 0;
    for(int k = len-2;k--;) {
```

```
            if ((unsigned char)curMsg[k] == 0xff) {
                if ((unsigned char)curMsg[k + 1] == 0xd8) {
                    offset = len - k;
                    memcpy((unsigned char *)recMsg,(unsigned char *)(curMsg + k),offset);
                    curMsg = recMsg + offset;
                    len = 0;
                    stat = 1;
                    mexPrintf("offset = %d¥n",offset);
                    break;
                }
            }
        }
    }
////////////////////////////////////////////////////////
    int totallen = 0;
    int k = 0;
    while(1) {
        len = Recv(Socket,curMsg,BUFMAX,0);
        if (len < 0) continue;
        curMsg -= offset;
        len += offset;
        offset = 0;
        for(k = 0;k < len - 1;k++) {
            if ((unsigned char)curMsg[k] == 0xff) {
                if ((unsigned char)curMsg[k + 1] == 0xd8) {
                    totallen = len - k;
                    memcpy((unsigned char *)recMsg,(unsigned char *)(curMsg + k),totallen);
                    curMsg = recMsg + totallen;
                    len = 0;
                    stat = 1;
                } else {
                    if(((unsigned char)curMsg[k + 1] == 0xd9)) {
                        if(stat == 0) continue;
                        totallen += (k + 2);
                        goto AAA;
                    }
                }
            }
        }
        curMsg += len;
        totallen += len;
    }
AAA:;
    int dims[2];
    dims[0]    = totallen;
    dims[1]    = 1;
    plhs[0] = mxCreateNumericArray(2, dims, mxUINT8_CLASS, mxREAL);
    char * matptr = (char *)mxGetPr(plhs[0]);
    memcpy((char *)matptr,(char *)recMsg,totallen);
    offset = len - (k + 2);
    memcpy((char *)recMsg,(char *)(curMsg + k + 2),offset);
    return;
}
```

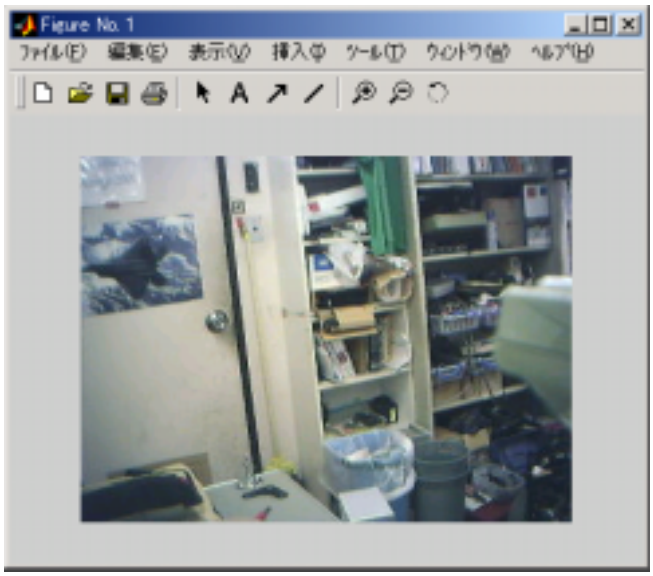
以上のスクリプトをまとめると、

samplempg2.m

```
import java.io.*;
import java.net.*;
import java.util.*;
PORT = 80;
HOST=”192.168.1.3;
PASS=’MDEyMzQ6NTY3ODkw’;
figure(1);set(1,'doublebuffer','on','Interruptible','on');
jpgstr=mhttp2mpgstr(HOST,PORT,PASS);
abc=imshow(jpgstr2img(jpgstr));
set(abc,'ButtonDownFcn','mclient4;');
try
    i=1;
```

```
    while(1)
        tic;jpgstr=mhttp2mpgstr(HOST,PORT,PASS);
        imgdat=jpgstr2img(jpgstr);
        toc,i=i+1
        set(abc,'CData',imgdat);
        drawnow;
    end
catch
    clear mex;
end
clear mex
```

となる。なお、ここでも同様に mclient4.m を使用する。
なお、この処理により速度は、平均で 0.3 秒程度まで連続してキャプチャーすることができた。



HandleGraphics によるボタンの実装

MATLAB の Handlegraphics を使うと、画面上にボタンなどを設置することができる。現状のスクリプトのままでも、画面上をマウスでクリックすることでカメラの向きを移動することができるが、ここでは、マウスのボタン以外にも画面上にボタンを設置し、操作できるようにスクリプトを改造する。方針としては、カメラの上下、左右の移動は、処理速度はそれほど要求されないため、C-mex 化せず、MATLAB-Java による実装にする。このようにすることで容易に、ズームなどの機能も追加することができるようになる。（ここでの KX-HCM1 では、ズーム機能はないが、）
MATLAB-Java 関連に関する関数は、以下に示す netcamctl 関数にまとめて記述する。

```
function netcamctrl(HOST,PORT,PASS,MOTION)
% MOTION = 'PanLeft';
% MOTION = 'PanRight';
% MOTION = 'TiltUp';
% MOTION = 'TiltDown';
import java.io.*;
import java.net.*;
import java.util.*;
aSocket=java.net.Socket(HOST,PORT);
autoflush = 1;
out     = java.io.PrintWriter(getOutputStream(aSocket), autoflush);
aaa=['GET /nphControlCamera?Direction=',MOTION,' HTTP/1.0'];
out.println(aaa);
aaa=['Authorization: Basic ',PASS];
out.println(aaa);
```

```
out.println('Host: localhost:8080');
out.println('Connection: Close');
out.println('');
close(aSocket);
```

また、マウスクリックによりカメラの向きを制御するスクリプトを、

mclient5.m

```
pos=get(gca,'CurrentPoint');
pos=pos(1,1:2);
psize=get(gca,'PlotBoxAspectRatio');
psize=psize(1:2);
direction=(psize/2-pos);
if direction(1)>0
    motion1 = 'PanLeft';
else
    motion1 = 'PanRight';
end
if direction(2)>0
    motion2 = 'TiltUp';
else
    motion2 = 'TiltDown';
end
if abs(direction(1))>30
    netcamctl(HOST,PORT,PASS,motion1);
end
if abs(direction(2))>30
    netcamctl(HOST,PORT,PASS,motion2);
end
```

それに、本体のプログラムを samplempg2.m

```
PORT = 80;
HOST="192.168.1.3;
PASS=' MDEyMzQ6NTY3ODkw';
figure(1);set(1,'doublebuffer','on','Interruptible','on');
jpgstr=mhttp2mpgstr(HOST,PORT,PASS);
abc=imshow(jpgstr2img(jpgstr));
set(abc,'ButtonDownFcn','mclient5;');
ui1=uicontrol('Position',[0 140 45 20],'string','Left','callback','netcamctl(HOST,PORT,PASS,"PanLeft");');
ui2=uicontrol('Position',[370 140 45 20],'string','Right','callback','netcamctl(HOST,PORT,PASS,"PanRight");');
ui3=uicontrol('Position',[190 270 50 20],'string','UP','callback','netcamctl(HOST,PORT,PASS,"TiltUp");');
ui4=uicontrol('Position',[190 5 50 20],'string','Down','callback','netcamctl(HOST,PORT,PASS,"TiltDown");');
ui5=uicontrol('Position',[0 5 50 20],'string','STOP','callback','motion = 0;');
motion=1;
try
    i=1;
    while(motion)
        tic;imgdat=jpgstr2img(mhttp2mpgstr(HOST,PORT,PASS));
        toc,i=i+1
        set(abc,'CData',imgdat);
        drawnow;
    end
catch
    clear mex;
end
clear mex
```

とする。以下に実行した画面を示す。

Vcapg4ncam の作成（mhttp2mpgstr 関数と、jpgstr2img 関数の統合）

これまでで一応、動作確認ができたので、今度は、mhttp2mpgstr 関数と jpgstr2img 関数を統合する。これによりデータのやり取りにおけるオーバーヘッドを減らし更なる高速化を図る。
以下に、vcapg2ncam.cpp を示す。

```
// mex vcap4ncam.cpp
#include "mex.h"
#include <string.h>
#include <setjmp.h>
#include <ws2spi.h>

extern "C" {
    #define XMD_H
    #undef FAR
    #include "jpeglib.h"
}
#pragma comment(lib, "libjpeg.lib")
#pragma comment(lib, "ws2_32.lib")
#define BUFMAX 65535
static char sendMsg[BUFMAX];
static char *recMsg;
char *curMsg;
static int totallen = 0;
static mxArray *ReadRgbJPEG(j_decompress_ptr cinfoPtr);

struct my_error_mgr {
    struct jpeg_error_mgr pub;  /* "public" fields */
    jmp_buf setjmp_buffer;   /* for return to caller */
};

typedef struct my_error_mgr *my_error_ptr;

typedef struct {
    struct jpeg_source_mgr pub;    /* public fields */
} my_source_mgr;

typedef my_source_mgr * my_src_ptr;


static void my_error_exit (j_common_ptr cinfo) {
    my_error_ptr myerr = (my_error_ptr) cinfo->err;
    (*cinfo->err->output_message) (cinfo);
    longjmp(myerr->setjmp_buffer, 1);
```

```
}
static void my_output_message (j_common_ptr cinfo) {
    char buffer[JMSG_LENGTH_MAX];
    (*cinfo->err->format_message) (cinfo, buffer);
    mexWarnMsgTxt(buffer);
}

static unsigned long gethostaddress(LPSTR szBuff) {
    PHOSTENT phe;
    struct in_addr in;
    phe = gethostbyname(szBuff);
    if(phe == NULL) exit(1);
    memcpy(&in,phe->h_addr,sizeof(in));
    mexPrintf("Host name : %s¥n",phe->h_name);
    mexPrintf("IP address: %s¥n",inet_ntoa(in));
    return *((unsigned long *)phe->h_addr);
}
int Recv(SOCKET s, char *inbuf, int len, int flags) {
    int rc = 0;
    int rc1;
    fd_set rset;
    struct timeval rtmout;
    for(;;) {
        rc = recv(s, inbuf, len, flags);
        if(rc > 0) return rc;
        if(WSAGetLastError() == WSAEWOULDBLOCK) {
            FD_ZERO(&rset);
            FD_SET(s, &rset);
            rtmout.tv_sec = 5;
            rtmout.tv_usec = 0;
            rc1 = select(FD_SETSIZE, &rset, NULL, NULL, &rtmout);
            if(rc1 == 0) {
                mexPrintf("select RECV TIMEOUT:¥n");
                return -1;
            }
            if(rc1 == SOCKET_ERROR) {
                mexPrintf("select RECV ERROR: err=%d¥n", WSAGetLastError());
                return -1;
            }
            if(!FD_ISSET(rset.fd_array[0], &rset)) {
                mexPrintf("Select RECV wakes but FD_ISSET() FAILS¥n");
                return -1;
            }
            continue;
        }
        return -1;
    }
}


METHODDEF(void) init_source (j_decompress_ptr cinfo) {
    my_src_ptr src = (my_src_ptr) cinfo->src;
}
METHODDEF(boolean) fill_input_buffer (j_decompress_ptr cinfo) {
    my_src_ptr src = (my_src_ptr) cinfo->src;
    src->pub.next_input_byte = (unsigned char *)recMsg;
    src->pub.bytes_in_buffer = (long)totallen;
    return TRUE;
}
METHODDEF(void) skip_input_data (j_decompress_ptr cinfo, long num_bytes) {
```

```
    my_src_ptr src = (my_src_ptr) cinfo->src;
    if (num_bytes > 0) {
        while (num_bytes > (long) src->pub.bytes_in_buffer) {
            num_bytes -= (long) src->pub.bytes_in_buffer;
            (void) fill_input_buffer(cinfo);
        }
        src->pub.next_input_byte += (size_t) num_bytes;
        src->pub.bytes_in_buffer -= (size_t) num_bytes;
    }
}

METHODDEF(void) term_source (j_decompress_ptr cinfo){}

void jpeg_string_src(j_decompress_ptr cinfo) {
    my_src_ptr src;
    if (cinfo->src == NULL) { /* first time for this JPEG object? */
        cinfo->src = (struct jpeg_source_mgr *)(*cinfo->mem->alloc_small) ((j_common_ptr) cinfo,
JPOOL_PERMANENT,sizeof(my_source_mgr));
        src = (my_src_ptr) cinfo->src;
    }
    src = (my_src_ptr) cinfo->src;
    src->pub.init_source = init_source;
    src->pub.fill_input_buffer = fill_input_buffer;
    src->pub.skip_input_data = skip_input_data;
    src->pub.resync_to_restart = jpeg_resync_to_restart; /* use default method */
    src->pub.term_source = term_source;
    src->pub.bytes_in_buffer = 0; /* forces fill_input_buffer on first read */
    src->pub.next_input_byte = NULL; /* until buffer loaded */
    return;
}

static mxArray * ReadRgbJPEG(j_decompress_ptr cinfoPtr) {
    long i,j,row_stride;
    int dims[3];                    /* For the call to mxCreateNumericArray */
    mxArray *img;
    JSAMPARRAY buffer;
    int current_row;
    uint8_T *pr_red, *pr_green, *pr_blue;
    row_stride = cinfoPtr->output_width * cinfoPtr->output_components;
    buffer = (*cinfoPtr->mem->alloc_sarray)((j_common_ptr) cinfoPtr, JPOOL_IMAGE, row_stride, 1);
    dims[0]   = cinfoPtr->output_height;
    dims[1]   = cinfoPtr->output_width;
    dims[2]   = 3;
    img = mxCreateNumericArray(3, dims, mxUINT8_CLASS, mxREAL);
    pr_red    = (uint8_T *) mxGetData(img);
    pr_green = pr_red + (dims[0]*dims[1]);
    pr_blue   = pr_red + (2*dims[0]*dims[1]);
    while (cinfoPtr->output_scanline < cinfoPtr->output_height) {
        current_row = cinfoPtr->output_scanline; /* Temp var won't get ++'d */
        jpeg_read_scanlines(cinfoPtr, buffer,1); /*   by jpeg_read_scanlines */
        for (i=0;i<cinfoPtr->output_width;i++) {
            j=(i)*cinfoPtr->output_height+current_row;
            pr_red[j]    = buffer[0][i*3+0];
            pr_green[j] = buffer[0][i*3+1];
            pr_blue[j]   = buffer[0][i*3+2];
        }
    }
    return img;
}
```

```
static SOCKET Socket = NULL;
static int start = 0;
void closeall(void) {
    mxFree(recMsg);
    closesocket(Socket);
    WSACleanup();
    start = 0;
    mexPrintf("¥nClear vcapg4ncam by Gerox(c) 2003¥n");
    mexPrintf("This software is using IJG library for JPEG decompression¥n");
}
static struct sockaddr_in clientName;
static int number;
static unsigned long ul = 1;
static WORD wVerReq;
static WSADATA wsadata;
static int recvbufsz = BUFMAX;
static int sendbufsz = BUFMAX;
static int offset = 0;
void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[]) {
    int len = 0;
    int stat = 0;
    if(start == 0) {
        if(nrhs != 3) {
            mexPrintf(" Image captureing tool for Panasonic Network camera¥n");
            mexPrintf(" KX-HCM1, KX-HCM2,KX-HCM130,KX-HCM170,KX-HCM180¥n");
            mexPrintf("Usage: rgbimg = vcapg4ncam(HOST,PORT,ENCRYPTPASS)¥n");
            mexPrintf("Example: rgbimg = vcapg4ncam('www.ikko.k.hosei.ac.jp',80,'MTIzNDo1Njc4OQ==');¥n");
            mexPrintf("Example: rgbimg = vcapg4ncam('192.168.1.3',80,'MTIzNDo1Njc4OQ==');¥n");
            mexPrintf("----------------------------------------¥n");
            mexPrintf("%s    by Gerox(c) Build %s¥n",__FILE__,__DATE__);
            mexPrintf("----------------------------------------¥n");
            return;
        }
        wVerReq = MAKEWORD(2,0);
        if(WSAStartup(wVerReq,&wsadata) != 0) return;
        recMsg = (char *)mxCalloc(BUFMAX,sizeof(char));
        mexMakeMemoryPersistent(recMsg);
        mexAtExit(closeall);
        start = 1;
        int hostnamelen = mxGetM(prhs[0])*mxGetN(prhs[0])+1;
        char hostname[255];
        mxGetString(prhs[0],hostname,hostnamelen);
        int portnum = (int) mxGetScalar(prhs[1]);
        char password[255];
        int passwordlen = mxGetM(prhs[2])*mxGetN(prhs[2])+1;
        mxGetString(prhs[2],password,passwordlen);

        Socket = WSASocket(AF_INET,SOCK_STREAM,0,NULL,0,WSA_FLAG_OVERLAPPED);
        if(Socket == INVALID_SOCKET) {
            mexPrintf("socket FAILED: err=%d¥n", WSAGetLastError());
            return;
        }
        memset(&clientName,0x00,sizeof(clientName));
        clientName.sin_family = AF_INET;
        clientName.sin_addr.s_addr = gethostaddress(hostname);
        clientName.sin_port = htons((unsigned short)portnum);
        if(connect(Socket,(struct sockaddr FAR *)&clientName,sizeof(clientName)) < 0) {
            mexPrintf("Connection fail %d¥n", WSAGetLastError());
```

```
            return;
        }
        number = sizeof(number);
        if(setsockopt(Socket,getprotobyname("tcp")->p_proto,TCP_NODELAY,(const char
*)&number,sizeof(number))) {
            mexPrintf("setsockopt FAILED: err=%d¥n", WSAGetLastError());
            return;
        }
        ul = 1;
        if(ioctlsocket(Socket, FIONBIO, &ul )) {
            mexPrintf("ioctlsocket FIONBIO Child: FAILED err=%d¥n",WSAGetLastError());
            return;
        }
        recvbufsz = BUFMAX;
        if(setsockopt(Socket,SOL_SOCKET,SO_RCVBUF,(char *)&recvbufsz,sizeof(recvbufsz))) {
            mexPrintf("setsockopt(SO_RCVBUF %d FAILED: err=%d¥n",recvbufsz, WSAGetLastError());
            return;
        }
        sendbufsz = BUFMAX;
        if(setsockopt(Socket,SOL_SOCKET,SO_SNDBUF,(char *)&sendbufsz,sizeof(sendbufsz))) {
            mexPrintf("setsockopt(SO_SNDBUF %d FAILED: err=%d¥n",sendbufsz, WSAGetLastError());
            return;
        }
        sprintf(sendMsg,"GET /nphMotionJPEG?Resolution=320x240&Quality=Standard HTTP/1.0¥n¥r");
        if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
        sprintf(sendMsg,"Authorization: Basic %s¥n¥r",password);
        if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
        sprintf(sendMsg,"Connection: Close¥n¥r");
        if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
        sprintf(sendMsg,"¥n¥r¥n¥r");
        if(send(Socket,sendMsg,strlen(sendMsg),0) < 0) mexPrintf("send error %d¥n",WSAGetLastError());
        curMsg = recMsg;
    } else {
        stat = 0;
        curMsg = recMsg + offset;
        len = Recv(Socket,curMsg,BUFMAX-offset,0);
        curMsg -= offset;
        len += offset;
        offset = 0;
        for(int k = len-2;k--;) {
            if ((unsigned char)curMsg[k] == 0xff) {
                if ((unsigned char)curMsg[k + 1] == 0xd8) {
                    offset = len - k;
                    memcpy((unsigned char *)recMsg,(unsigned char *)(curMsg + k),offset);
                    curMsg = recMsg + offset;
                    len = 0;
                    stat = 1;
                    break;
                }
            }
        }
    }
////////////////////////////////////////////////////////////////
    totallen = 0;
    int k = 0;
    while(1) {
        Sleep(0);
        len = Recv(Socket,curMsg,BUFMAX,0);
        if (len < 0) continue;
        curMsg -= offset;
```

```
        len += offset;
        offset = 0;
        for(k = 0;k < len - 1;k++) {
            if ((unsigned char)curMsg[k] == 0xff) {
                if ((unsigned char)curMsg[k + 1] == 0xd8) {
                    totallen = len - k;
                    memcpy((unsigned char *)recMsg,(unsigned char *)(curMsg + k),totallen);
                    curMsg = recMsg + totallen;
                    len = 0;
                    stat = 1;
                } else {
                    if(((unsigned char)curMsg[k + 1] == 0xd9)) {
                        if(stat == 0) continue;
                        totallen += (k + 2);
                        goto AAA;
                    }
                }
            }
        }
        curMsg += len;
        totallen += len;
    }
AAA:;
////////////////////////////////
    struct jpeg_decompress_struct cinfo;
    struct my_error_mgr jerr;
    cinfo.err = jpeg_std_error(&jerr.pub);
    jerr.pub.output_message = my_output_message;
    jerr.pub.error_exit = my_error_exit;
    if(setjmp(jerr.setjmp_buffer)) {
        jpeg_destroy_decompress(&cinfo);
        closeall();
        return;
    }
////////////////////////////
    mxArray *outArray;
    jpeg_create_decompress(&cinfo);
    jpeg_string_src(&cinfo);
    jpeg_read_header(&cinfo, TRUE);
    jpeg_start_decompress(&cinfo);
    outArray = ReadRgbJPEG(&cinfo);
    jpeg_finish_decompress(&cinfo);
    jpeg_destroy_decompress(&cinfo);
    plhs[0]=outArray;
    offset = len - (k + 2);
    memcpy((char *)recMsg,(char *)(curMsg + k + 2),offset);
    return;
}
```

またこれらを統合し動作させるスクリプトを samplempg3.m を以下に示す。

```
PORT = 80;
HOST=”192.168.1.3;
PASS=’ MDEyMzQ6NTY3ODkw’;
figure(1);set(1,'doublebuffer','on','Interruptible','on');
abc=imshow(vcapg4ncam(HOST,PORT,PASS));
set(abc,'ButtonDownFcn','mclient4;');
ui1=uicontrol('Position',[0 140 45 20],'string','Left','callback','netcamctl(HOST,PORT,PASS,"PanLeft");');
ui2=uicontrol('Position',[370 140 45 20],'string','Right','callback','netcamctl(HOST,PORT,PASS,"PanRight");');
ui3=uicontrol('Position',[190 270 50 20],'string','UP','callback','netcamctl(HOST,PORT,PASS,"TiltUp");');
ui4=uicontrol('Position',[190 5 50 20],'string','Down','callback','netcamctl(HOST,PORT,PASS,"TiltDown");');
ui5=uicontrol('Position',[0 5 50 20],'string','STOP','callback','motion = 0;');
```

```
motion=1;
try
    i=1;
    while(motion)
        tic;imgdat=vcapg4ncam(HOST,PORT,PASS);
        toc,i=i+1
        set(abc,'CData',imgdat);
        drawnow;
    end
catch
    clear mex;
end
clear mex
```

ネットワークによる速度の影響のほうが大きいため、サンプル速度的には、ほとんど変わらないが、平均して１枚あたり 0.3 秒程度でアップデートすることができる。


Figure No. 1
UP
Left
Right
STOP
Down