JPEGについて

JPEGアルゴリズムの確認

[Reference]
http://en.wikipedia.org/wiki/Jpeg

2006/10/21 平野拓一（東京工業大学）
2006/10/21 Takuichi Hirano (Tokyo Institute of Technology)

BMPファイルの読み込み

```
In[1]:= SetDirectory["d:/hira2/public_html/hobby/edu/jpeg"]
Out[1]= d:\hira2\public_html\hobby\edu\jpeg

        SetDirectory["d:/Home/takuichi/hira2/public_html/hobby/edu/jpeg"]

In[2]:= g = Import["CIMG0958.BMP"]
Out[2]= - Graphics -

In[3]:= Show[g]
```

```
Out[3]= - Graphics -
```

InputForm[g]

```
In[4]:= ImgList = g[[1, 1]];  (* {R,G,B} の2次元リスト *)

        ImgListRed = Map[(#[[1]] &), ImgList, {2}];  (* R (0-255) の2次元リスト *)
        ImgListGreen = Map[(#[[2]] &), ImgList, {2}];  (* G (0-255) の2次元リスト *)
        ImgListBlue = Map[(#[[3]] &), ImgList, {2}];  (* B (0-255) の2次元リスト *)

        nx = Length[ImgList[[1]]];
        ny = Length[ImgList];
```

In[10]:=
```
MakeRGBRasterImage[ImgListRed_, ImgListGreen_, ImgListBlue_] := Module[{nx, ny},
   nx = Length[ImgListRed[[1]]];
   ny = Length[ImgListRed];
   Graphics[Raster[Table[{ImgListRed[[j, i]],
       ImgListGreen[[j, i]], ImgListBlue[[j, i]]}, {j, 1, ny}, {i, 1, nx}],
     {{0, 0}, {nx, ny}}, {0, 255}, ColorFunction → RGBColor]]
  ];
```

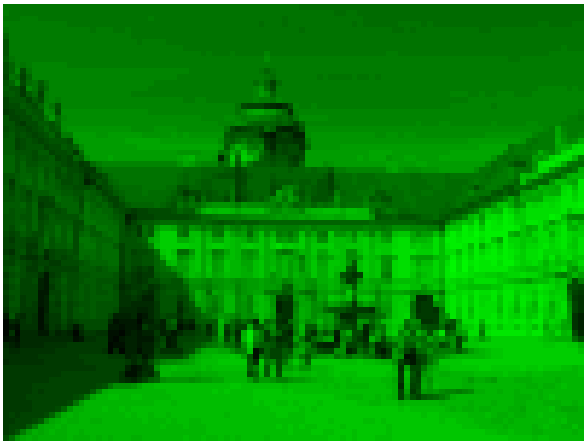
In[11]:=
```
Show[MakeRGBRasterImage[ImgListRed, ImgListGreen, ImgListBlue],
 AspectRatio → Automatic]
```

Out[11]= - Graphics -

In[12]:=
```
ZeroTable = Table[0, {n, 0, ny - 1}, {m, 0, nx - 1}];
Print["**** Red Component ****"];
Show[MakeRGBRasterImage[ImgListRed, ZeroTable, ZeroTable],
  AspectRatio → Automatic];
Print["**** Green Component ****"];
Show[MakeRGBRasterImage[ZeroTable, ImgListGreen, ZeroTable],
  AspectRatio → Automatic];
Print["**** Blue Component ****"];
Show[MakeRGBRasterImage[ZeroTable, ZeroTable, ImgListBlue],
  AspectRatio → Automatic];
```

**** Red Component ****

**** Green Component ****



**** Blue Component ****

(R,G,B) → (Y, Cb, Cr)
JPEGではRGBでなく、YCbCr(またはYUVとも言われる)でそれぞれの成分を処理する。YUVはテレビ放送のPAL方式で採用されている色表現。

http://en.wikipedia.org/wiki/YCbCr

The Y component represents the brightness of a pixel
The Cb and Cr components together represent the chrominance

The human eye can see more detail in the Y component than in Cb and Cr. Using this knowledge, encoders can be designed to compress images more efficiently.

In[19]:=
```
YCbCrFromRGB[{r_, g_, b_}] := {0.299 * r + 0.587 * g + 0.114 * b,
   128 - 0.168736 * r - 0.331264 * g + 0.5 * b,
   128 + 0.5 * r - 0.418688 * g - 0.081312 * b};
RGBFromYCbCr[{y_, cb_, cr_}] :=
  {-179.45579155073017` - 1.2175717664097974` *^-6 * cb + 1.4019995886573404` * cr + y,
   135.45879483222961` - 0.3441356788389385` * cb - 0.7141361555818125` * cr + y,
   -226.81606046360054` + 1.7720000660738162` * cb + 4.062980628562637` *^-7 * cr + y};
```

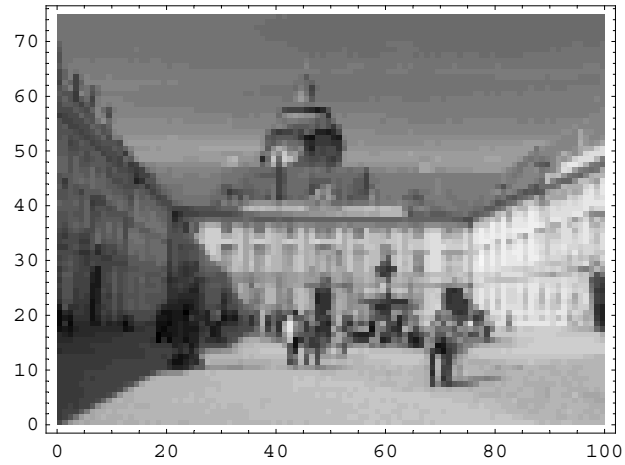
In[21]:=
```
lis = Map[(YCbCrFromRGB[#] &), ImgList, {2}];
ImgListY = Map[(#[[1]] &), lis, {2}];  (* Yの2次元リスト *)
ImgListCb = Map[(#[[2]] &), lis, {2}];  (* Cbの2次元リスト *)
ImgListCr = Map[(#[[3]] &), lis, {2}];  (* Crの2次元リスト *)

ImgListY = ImgListY - 128;  (* Yは128を引いて-128-127 の範囲にする *)
Null
```

```
General::spell1 :
```
スペル間違いの可能性があります．新規シンボル"ImgListY"はすでにあるシンボル"ImgList"に似ています． 詳細

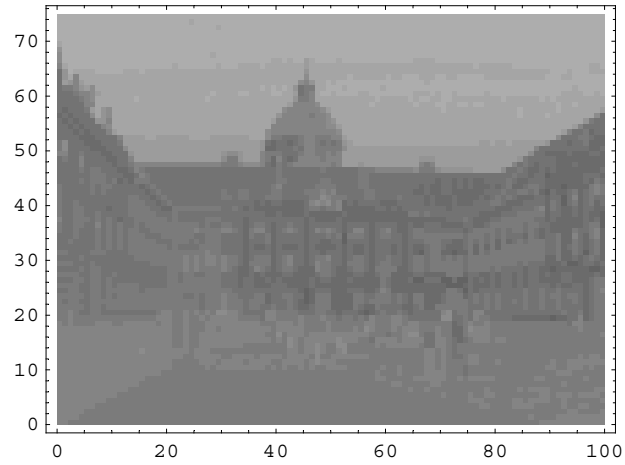
```
General::spell1 :
```
スペル間違いの可能性があります．新規シンボル"ImgListCr"はすでにあるシンボル"ImgListCb"に似ています． 詳細

In[27]:=
```
Print["**** Y Component ****"];
ListDensityPlot[ImgListY,
  AspectRatio → Automatic,
  PlotRange → {-128, 127},
  Mesh → False];
Print["**** Cb Component ****"];
ListDensityPlot[ImgListCb,
  AspectRatio → Automatic,
  PlotRange → {0, 255},
  Mesh → False];
Print["**** Cr Component ****"];
ListDensityPlot[ImgListCr,
  AspectRatio → Automatic,
  PlotRange → {0, 255},
  Mesh → False];
```

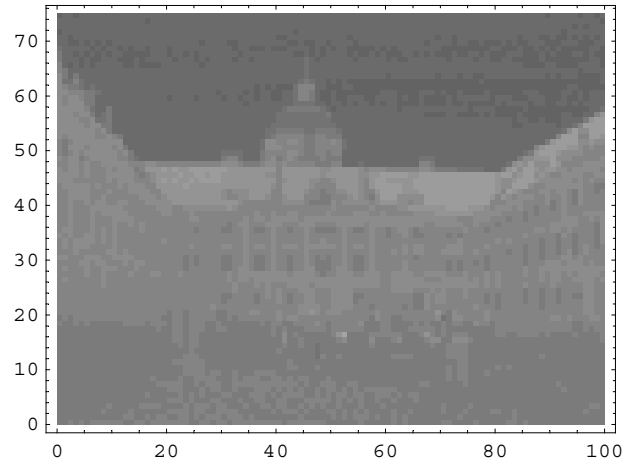
**** Y Component ****



**** Cb Component ****



**** Cr Component ****



DCT (Discrete Cosine Transform; **離散コサイン変換**)
IDCT (Inverse Discrete Cosine Transform; **逆離散コサイン変換**)

２次元リストにDCTを行う関数の定義

[参考]
貴家仁志：よくわかるディジタル画像処理、CQ**出版社、**p.123

In[33]:=
```
DCT[lis_] := Module[{c, g, nn},
   c[p_] := If[p == 0, 1/Sqrt[2], 1];
   nn = Length[lis];
   g[u_, v_] := 2/nn * c[u] * c[v] * Sum[lis[[m + 1, n + 1]] *
       Cos[(2 * m + 1) * u * π/(2 * nn)] * Cos[(2 * n + 1) * v * π/(2 * nn)], {m, 0, nn - 1}, {n, 0, nn - 1}];
   Table[g[u, v], {u, 0, nn - 1}, {v, 0, nn - 1}]
  ];
IDCT[lis_] := Module[{c, f, nn},
   c[p_] := If[p == 0, 1/Sqrt[2], 1];
   nn = Length[lis];
   f[m_, n_] := 2/nn * Sum[c[u] * c[v] * lis[[u + 1, v + 1]] *
       Cos[(2 * m + 1) * u * π/(2 * nn)] * Cos[(2 * n + 1) * v * π/(2 * nn)], {u, 0, nn - 1}, {v, 0, nn - 1}];
   Table[f[u, v], {u, 0, nn - 1}, {v, 0, nn - 1}]
  ];
```

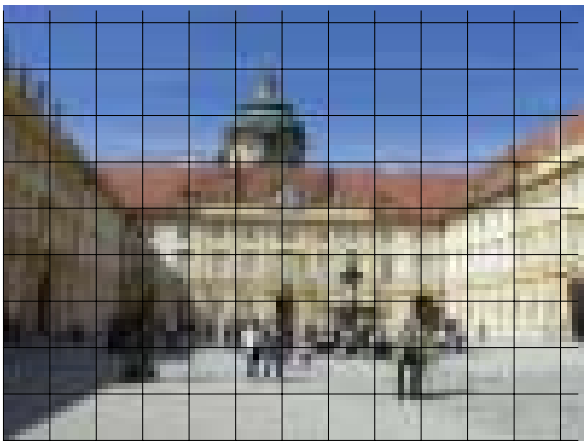
General::spell1 : **スペル間違いの可能性があります．新規シンボル**"IDCT"**はすでにあるシンボル**"DCT"**に似ています．** **詳細**

量子化テーブル

In[35]:=
```
qty = {{16, 11, 10, 16, 24, 40, 51, 61},
       {12, 12, 14, 19, 26, 58, 60, 55},
       {14, 13, 16, 24, 40, 57, 69, 56},
       {14, 17, 22, 29, 51, 87, 80, 62},
       {18, 22, 37, 56, 68, 109, 103, 77},
       {24, 35, 55, 64, 81, 104, 113, 92},
       {49, 64, 78, 87, 103, 121, 120, 101},
       {72, 92, 95, 98, 112, 100, 103, 99}};  (* Y成分用 *)

qtcbcr = {{17, 18, 24, 47, 99, 99, 99, 99},
          {18, 21, 26, 66, 99, 99, 99, 99},
          {24, 26, 56, 99, 99, 99, 99, 99},
          {47, 66, 99, 99, 99, 99, 99, 99},
          {99, 99, 99, 99, 99, 99, 99, 99},
          {99, 99, 99, 99, 99, 99, 99, 99},
          {99, 99, 99, 99, 99, 99, 99, 99},
          {99, 99, 99, 99, 99, 99, 99, 99}};  (* Cb,Cr成分用 *)

s = 0.5;  (* quality *)
```

8 x 8の部分画像に分解

In[38]:=
```
ix = Ceiling[nx / 8];
jy = Ceiling[ny / 8];
Do[
  j0 = If[j < jy, (j - 1) * 8 + 1, ny - 8 + 1];
  Do[
   i0 = If[i < ix, (i - 1) * 8 + 1, nx - 8 + 1];
   img8x8RGB[i, j] = Take[ImgList, {j0, j0 + 7}, {i0, i0 + 7}];
   , {i, 1, ix}
  ];
  , {j, 1, jy}
 ];
Show[{MakeRGBRasterImage[ImgListRed, ImgListGreen, ImgListBlue],
   Graphics[{RGBColor[0, 0, 0], Table[Line[{{0, (j - 1) * 8}, {nx - 1, (j - 1) * 8}}],
      {j, 1, jy}], Table[Line[{{(i - 1) * 8, 0}, {(i - 1) * 8, ny - 1}}], {i, 1, ix}]}]},
  AspectRatio → Automatic];
```



In[42]:=
```
CODEC8x8Block[i_, j_] := Module[{FImgListY, FImgListCb, FImgListCr,
    FImgListY2, FImgListCb2, FImgListCr2, FImgListY3, FImgListCb3,
    FImgListCr3, IFFImgListY, IFFImgListCb, IFFImgListCr, ImgList2, k, l},
   ImgList = img8x8RGB[i, j];   (* {R,G,B} の2次元リスト *)

   ImgListRed = Map[(#[[1]] &), ImgList, {2}];   (* R (0-255) の2次元リスト *)
   ImgListGreen = Map[(#[[2]] &), ImgList, {2}];   (* G (0-255) の2次元リスト *)
   ImgListBlue = Map[(#[[3]] &), ImgList, {2}];   (* B (0-255) の2次元リスト *)

   lis = Map[(YCbCrFromRGB[#] &), ImgList, {2}];   (* YCbCrに変換 *)
   ImgListY = Map[(#[[1]] &), lis, {2}];   (* Yの2次元リスト *)
   ImgListCb = Map[(#[[2]] &), lis, {2}];   (* Cbの2次元リスト *)
   ImgListCr = Map[(#[[3]] &), lis, {2}];   (* Crの2次元リスト *)

   ImgListY = ImgListY - 128;   (* Yは128を引いて-128-127 の範囲にする *)

   (* DCT *)
   FImgListY = DCT[ImgListY];
   FImgListCb = DCT[ImgListCb];
   FImgListCr = DCT[ImgListCr];

   (* Quantization *)
   FImgListY2 = Round[FImgListY / (qty / s)];
   FImgListCb2 = Round[FImgListCb / (qtcbcr / s)];
   FImgListCr2 = Round[FImgListCr / (qtcbcr / s)];

   (* Zigzag Scan, Entropy Encoding, Generate JPEG File *)

   (* Inverse Quantization *)
   FImgListY3 = Round[FImgListY2 * (qty / s)];
   FImgListCb3 = Round[FImgListCb2 * (qtcbcr / s)];
   FImgListCr3 = Round[FImgListCr2 * (qtcbcr / s)];

   (* Inverse DCT *)
   IFFImgListY = IDCT[FImgListY3];
   IFFImgListCb = IDCT[FImgListCb3];
   IFFImgListCr = IDCT[FImgListCr3];

   IFFImgListY = IFFImgListY + 128;   (* Yは128を足して0-255 の範囲にする *)
   ImgList2 = Table[{IFFImgListY[[k, l]],
      IFFImgListCb[[k, l]], IFFImgListCr[[k, l]]}, {k, 1, 8}, {l, 1, 8}];

   (* RGBに変換 *)
   Map[(RGBFromYCbCr[#] &), ImgList2, {2}]
  ];
```

General::spell1 :
スペル間違いの可能性があります．新規シンボル"FImgListY"はすでにあるシンボル"ImgListY"に似ています． 詳細

General::spell1 :
スペル間違いの可能性があります．新規シンボル"FImgListCb"はすでにあるシンボル"ImgListCb"に似ています． 詳細

General::spell : スペル間違いの可能性があります．新規シンボル"FImgListCr"はすでにあるシンボル {FImgListCb, ImgListCr}に似ています． 詳細

General::spell1 :
スペル間違いの可能性があります．新規シンボル"FImgListCr2"はすでにあるシンボル"FImgListCb2"に似ています． 詳細

General::stop : 計算中，General::spell1のこれ以上の出力は表示されません． 詳細

In[43]:=
```
k = 1;
kk = ix * jy;
Do[
 Do[
  ImgList2[i, j] = CODEC8x8Block[i, j];

  If[Mod[k, 10] == 0, Print[k, "/", kk]];
  k++;
  , {i, 1, ix}
 ];
 , {j, 1, jy}
];
```

```
10/130

20/130

30/130

40/130

50/130

60/130

70/130

80/130

90/130

100/130

110/130

120/130

130/130
```

画像を表示

8 x 8のブロックに分けて重なった部分を切り取る

In[46]:=
```
Do[
  Do[
    ImgList3[i, j] = ImgList2[i, j];
    , {i, 1, ix}
   ];
  , {j, 1, jy}
 ];

nxcut = If[Mod[nx, 8] ≠ 0, 8 - Mod[nx, 8], 0];
Do[
  ImgList3[ix, j] = Map[(Drop[#, nxcut]) &, ImgList2[ix, j]];
  , {j, 1, jy}
 ];

nycut = If[Mod[ny, 8] ≠ 0, 8 - Mod[ny, 8], 0];
Do[
  ImgList3[i, jy] = Drop[ImgList2[i, jy], nycut];
  , {i, 1, ix}
 ];
```

General::spell1 : スペル間違いの可能性があります．新規シンボル"nycut"はすでにあるシンボル"nxcut"に似ています． 詳細

結果を表示

In[51]:=
```
ImgList4 = Flatten[Table[Table[Flatten[Table[ImgList3[i, j][[k]], {i, 1, ix}], 1],
     {k, 1, Length[ImgList3[1, j]]}], {j, 1, jy}], 1];
ImgListR2 = Map[(#[[1]] &), ImgList4, {2}];   (* R (0-255) の2次元リスト *)
ImgListG2 = Map[(#[[2]] &), ImgList4, {2}];   (* G (0-255) の2次元リスト *)
ImgListB2 = Map[(#[[3]] &), ImgList4, {2}];   (* B (0-255) の2次元リスト *)
Show[MakeRGBRasterImage[ImgListR2, ImgListG2, ImgListB2],
 AspectRatio → Automatic];
```

General::spell1 :
スペル間違いの可能性があります．新規シンボル"ImgListR2"はすでにあるシンボル"ImgList2"に似ています． 詳細

General::spell :
スペル間違いの可能性があります．新規シンボル"ImgListG2"はすでにあるシンボル{ImgList2, ImgListR2}に似ています． 詳細

General::spell : スペル間違いの可能性があります．新規シンボル"ImgListB2"はすでにあるシンボル
{ImgList2, ImgListG2, ImgListR2}に似ています． 詳細