Evaluation value of digital still image
by using pixel color data, and image search as its application

HATADA Akinobu

1. Image Search in Internet

Image search is an application on Internet. It search image by text keyword. Search engine looking for keyword text from text that associated to image file on web site, and return image by way of text matching (Figure 1).
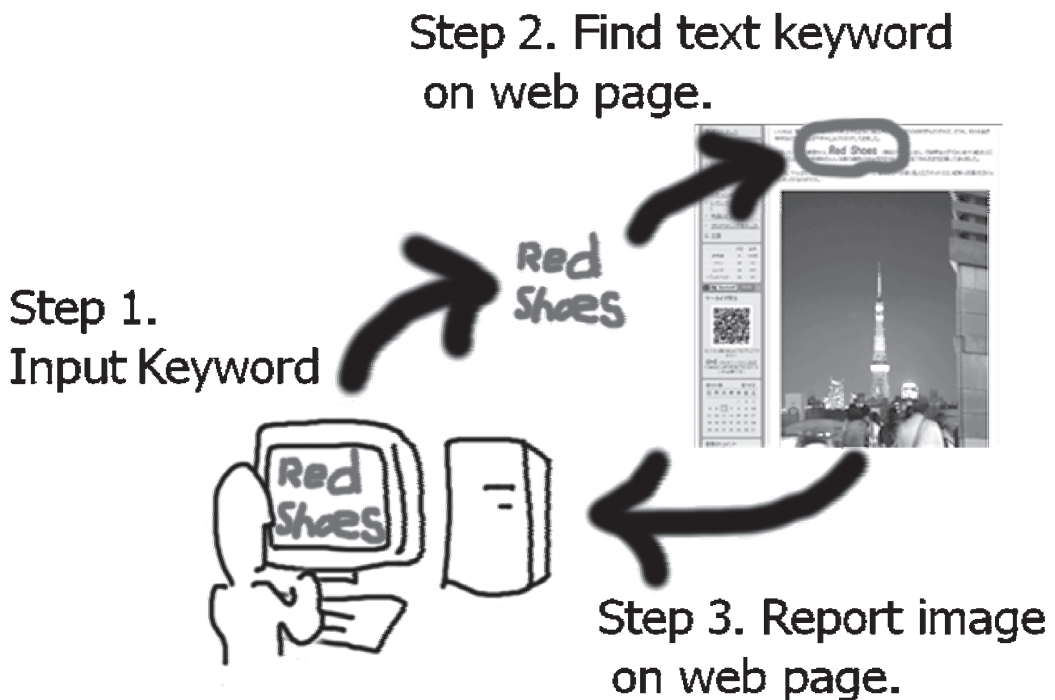

Figure 1. Image Search by Keyword Text

This method based on text searching systems. It means image search system do not refer image data itself. It might feel strange, but it is true. If someone wants to search similar image by image, it is not avairable.

2. Image Search by Image

What image search by image in this paper is searching image by image instead key word (Figure 2). It analyze image data and looking for image, not text. It also searches similar image rather than same image toward target image.
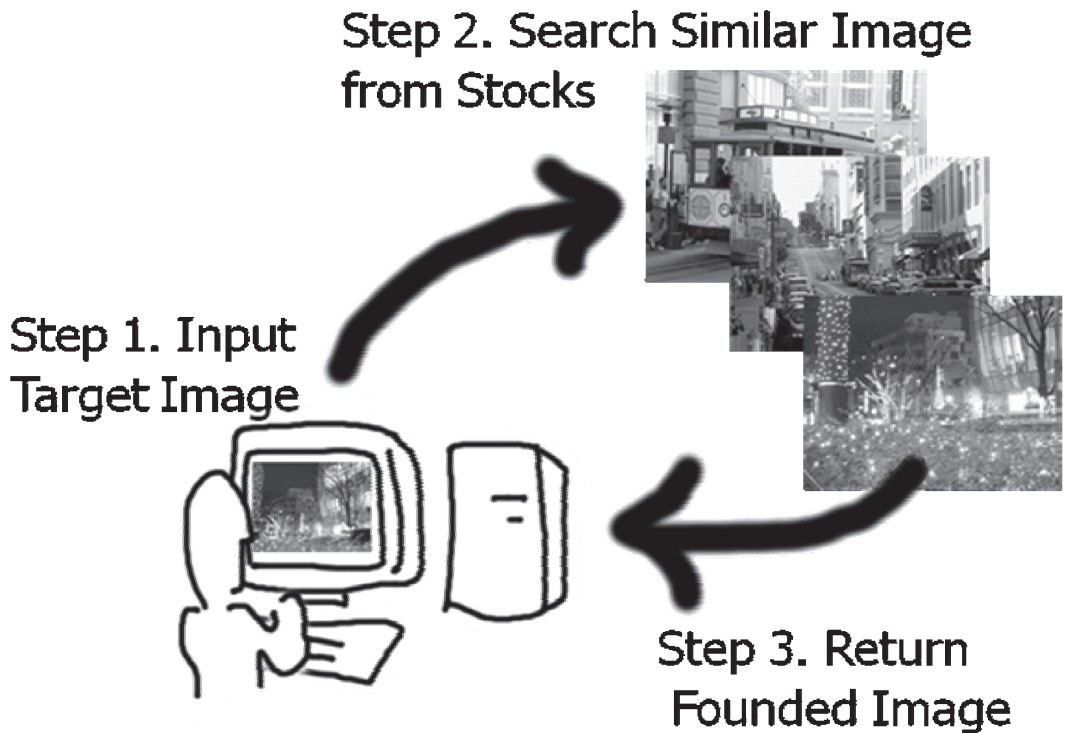

Figure 2. Image Search by Image

For looking for same image by image, CRC value, MD5 value or similar value might be used. A combination of those kind of image file evaluated value and vertical/horizontal image size, file size, time stamp and so on, would be enough to find same image. But it is not seem useful in practically and it is not provided as actual online sevices. Because it will easy to lose from search result by even rotate or zoom image.

3. Approach

For developing search system, we need to define what similar image is. As well known, human beings recognize image in several way. For instance, face recognition is done by specific brain part. This fact is confirmed by case report of some person destroyed some brain region and lost an avirity to recognize human face. He/she seems same to any people's face.

This paper report a method to searching similar image for street view. It means that is not avairable search some person or object in image. This algorithm use only color data in image pixels. Then definition of 'Similar Image' in this paper is 'Similar Color Pixel Included Image'.

This might seems strange, because it is not use any position, locate or edge information of image. But location data is not suitable to image searching purpose because it will easy to be a reson to lost image. For example, once lotate image, then it will be very difficult to recognize same image by position data (Figure 3).
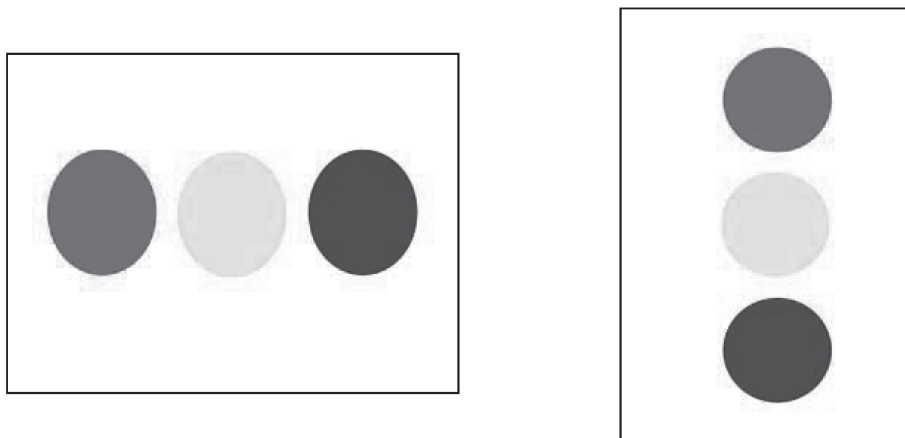

Figure 3. Sample Image—These are same, even if rotated.

4. Making Evaluating Value of Image

Image data evaluating value is made by follower step, 1) resize image to same size (1024 x 764 in this trial), 2) reduce number of color from 24 bit to 9 bit, 3) scan every pixel and count pixel number by each color, 4) count number of flood fill region. Then evaluating value would be generated. Sample image and evaluated values are shown at Figure 4.
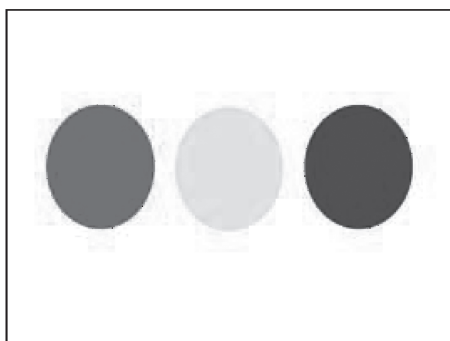
R	G	B	num_of_pix	num_of_isla
0	0	224	55035	1
224	0	0	55033	1
224	224	0	55041	1
224	224	224	619290	1

Figure 4. Sample Image and Evaluated Values.

For more practical example in Figure 5 and its evaluated values in Figure 6 and Figure 7. Figure 6 is a result of sort by pixel order and Figure 7 is a result of number of fill flood region. A sample code is listed Annex A-1.

Figure 5. More Practival Sample Images—Similar but different.

2つの画像のデータをピクセルの数で降順に並べ替えた											
IMG127	R	G	B	num_of_pix	num_of_island	IMG128	R	G	B	num_of_pix	num_of_isla
	32	0	0	126951	409		32	32	0	79559	134
	32	32	0	69491	654		128	96	0	69897	302
	64	64	32	34820	478		32	0	0	46524	76
	96	64	32	31887	753		96	64	32	43364	180
	128	96	0	31489	622		64	32	0	34293	211
	64	32	0	30909	683		96	64	0	33182	298
	0	0	0	24285	104		64	64	32	30132	123
	224	224	224	22446	366		96	96	0	27378	271
	128	96	32	21944	958		128	96	32	24218	224
	64	32	32	18432	606		224	224	224	22850	159
	96	64	0	17521	839		160	128	32	20903	198
	128	96	64	15763	449		160	128	0	15663	160
	160	128	32	13722	743		64	64	0	14783	132
	96	96	0	13230	503		64	32	32	14513	116
	32	32	32	11151	287		128	96	64	14410	105
	160	128	64	10631	371		160	128	64	10392	59
	224	192	96	9120	312		128	128	0	9120	74
	96	96	64	8583	224		160	96	32	8692	70
	224	224	192	8106	443		96	96	64	8444	53
	224	224	160	7603	403		96	96	32	8304	73
	64	64	0	7579	309		128	64	32	8225	64
	224	192	128	7572	344		224	224	192	7528	39
	64	96	224	6919	53		160	128	96	7447	43
	96	96	32	6883	365		0	0	0	6428	25

Figure 6. Evaluated Values—Sort by Pixel Order.

2つの画像のデータを島の数で降順に並べ替えた										
IMG127	R	G	B	num_of_pix	num_of_island	IMG128	R	G	B	num_of_pix	num_of_isla
	128	96	32	21944	958		128	96	0	69897	302
	96	64	0	17521	839		96	64	0	33182	298
	96	64	32	31887	753		96	96	0	27378	271
	160	128	32	13722	743		128	96	32	24218	224
	64	32	0	30909	683		64	32	0	34293	211
	32	32	0	69491	654		160	128	32	20903	198
	128	96	0	31489	622		96	64	32	43364	180
	64	32	32	18432	606		160	128	0	15663	160
	96	96	0	13230	503		224	224	224	22850	159
	64	64	32	34820	478		32	32	0	79559	134
	128	96	64	15763	449		64	64	0	14783	132
	224	224	192	8106	443		64	64	32	30132	123
	32	0	0	126951	409		64	32	32	14513	116
	224	224	160	7603	403		128	96	64	14410	105
	160	96	32	6275	396		32	0	0	46524	76
	160	128	64	10631	371		128	128	0	9120	74
	224	224	224	22446	366		96	96	32	8304	73
	96	96	32	6883	365		160	96	32	8692	70
	192	160	64	6645	348		128	64	32	8225	64
	224	192	128	7572	344		160	128	64	10392	59
	224	192	96	9120	312		96	96	64	8444	53
	64	64	0	7579	309		160	128	96	7447	43
	32	32	32	11151	287		224	224	192	7528	39
	160	128	0	5265	285		96	64	64	5914	39

Figure 7. Evaluated Values—Sort by Number of Fill-flood Region Order.

As see Figure 6 and Figure 7, top 15 or top 20 colors are easy to find both sides of evaluated values as far as order is different and some of them are not founded. This fact might indicate matching colors top some number of evaluated value list could be an algorithm for image search.

5. Image Search Algorithm

As study in previous section, color data based evaluating value might be used as image search index. Abstracted search procedure is 1) select color from target image evaluating values (target color), 2) search target color from evaluating value list of matching image and count it up, 3) select image if counted number at step 2 is larger than pre-set threshould. Actual search code is shown in section A-2 at Annex.
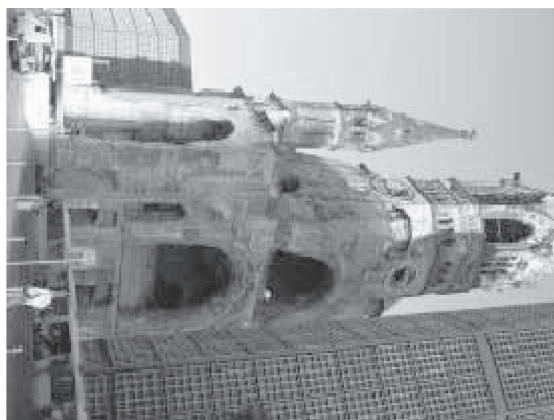

Figure 8. Sample Target Image

Study for this paper, use 288 number of street view image and did practice. One easy result will show in follower.

Figure 8 is a sample target image in this case. Search engine looking for similar image shown in Figure 8 from image stock. Search result is provides by pixel order and flood fill region order. Search result by pixel order is shown at Figure 9, flood fill region order is shown at Figure 10.
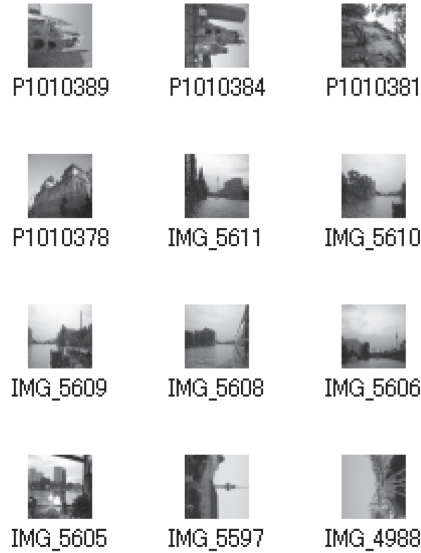

P1010389   P1010384   P1010381

P1010378   IMG_5611   IMG_5610

IMG_5609   IMG_5608   IMG_5606

IMG_5605   IMG_5597   IMG_4988

Figure 9. Search Result by Pixel Order

P1010389   P1010389   P1010389

P1010389   P1010389   P1010389

P1010384   P1010384   P1010384
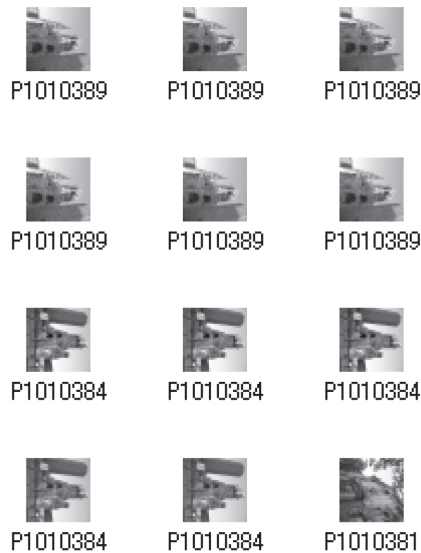
P1010384   P1010384   P1010381

Figure 10. Search Result by Flood-fill Region Order

As see Figure 9 and 10, search result is different by index data. In this case, flood fill region order seems better than pixel order data, but it is not gallanteed in any case. The point is only differences between those. Addtionally, both of them includes sure similally image. This fact could be a possibility of this method use for actual image searching application. In other hand, numbers of non-similary images seems hit as search result. This fact indicates that some rating method is required for actual searching application.

6. Remark

This paper introduces a method for evaluating digital still image of street view, and search algorithm as its application. Evaluating value is based on color data of each pixel of image and searching method is search color by ordering color pixel count or flood fill region. This scope might work well but it also picked up non-similar images. For better searching, it seems to require rating systems in search algorithm.

Rating is big issue at any searching system even in text search system. It often developed by ad-hoc way. But it seems required to more sure method by any proof in case of image search. For example, any colors do not make same impression to human begins. Some color makes strong impression but another might weak. Rating by color will be a future's issue.

Annex – Sample Code

A-1 Sample Pascal code for making evaluating value of digital still image file

```
procedure TMainForm.ColorPickup;
var
  h1,w1: Integer;
  i, j, k: Integer;
  r1,g1,b1: Integer;
  r2,g2,b2: Integer;
  r3,g3,b3: Byte;
  r4{,g4,b4}: Integer;
  r5,g5,b5: Integer;
  r6{,g6,b6}: Integer;
  total_valied_color : Integer;
  padding: Integer;
  S: string;
  P1 : PByteArray;
  P2 : PByteArray;
  P3 : Pointer;
  boh: BITMAPINFOHEADER;
  bit_count : Integer;
  size_image: Integer;
  color_used: Integer;
  image_offset: Integer;
  pixel_byte_depth: Integer;
  island_count: Integer;
  function check_one_pixel (axis_x, axis_y: Integer): Integer;
  var
    my_count: Integer;
  begin
    my_count:=0;
    k := image_offset + axis_y*(pixel_byte_depth*w1+padding) + axis_x*pixel_byte_depth;
    r3 := P2[k+2];
    g3 := P2[k+1];
    b3 := P2[k];
    r5 := r3;
    g5 := g3;
    b5 := b3;
    r5 := r5 div KU_COLOR_CHANEL_STEP;
    g5 := g5 div KU_COLOR_CHANEL_STEP;
```

```
  b5 := b5 div KU_COLOR_CHANEL_STEP;
 r3 := P1[k+2];
 r6 := r3;
 if (r6>0) and (r5=r2) and (g5=g2) and (b5=b2) then
 begin
  P1[k+2] := 0;
  P1[k+1] := 0;
  P1[k] := 0;
  Inc (island_count);
  Inc (my_count)
 end;
 check_one_pixel := 3; // different color
 if my_count > 0 then check_one_pixel := 0; // checked
 if r6=0 then check_one_pixel := 1; // already checked pixel
end;
procedure check_around_pixel;
var
 x1,y1, x0,x2,x, checked: Integer;
begin
    if ((i)>=0) and ((j)>=0) and ((i)<w1) and ((j)<h1) then
    begin
     checked := check_one_pixel (i, j);
     if checked > 0 then Exit;
     x1 := i;
     y1 := j;
     for x:=(i-1) downto 0 do
     begin
      checked := check_one_pixel(x,j);
      if checked > 0 then break;
     end;
     x0 := x-1;
     for x:=(i+1) to w1 do
     begin
      checked := check_one_pixel(x,j);
      if checked > 0 then break;
     end;
     x2 := x-1;
     for x:=x0 to x2 do
     begin
      i:=x;
```

```
      j:=y1-1;
      check_around_pixel;
      j:=y1+1;
      check_around_pixel
    end;
    i := x1;
    j := y1;
  end
end;
procedure add_island_record;
var
  rec: PRKUIsland;
begin
  if island_count <= trackIslandThreshold.Position then Exit;
  New(rec);
  rec^.pixel := island_count;
  rec^.x1 := i;
  rec^.y1 := j;
  rec^.next := color_list[r2,g2,b2].head;
  color_list[r2,g2,b2].head := rec;
end;
begin
  if (ImageHandle = 0) then Exit;
  P2 := GlobalLock(ImageHandle);
  CopyMemory(Addr(boh),P2,40);
  //i := boh.biSize; // always 40
  size_image := boh.biSizeImage;
  color_used := boh.biClrUsed;
  image_offset := 40 + color_used * 4; // offset of color palette
  w1 := boh.biWidth;
  h1 := boh.biHeight;
  bit_count := boh.biBitCount; // 24 if image color depth is 24bit
  pixel_byte_depth := bit_count div 8;
  padding := (w1*pixel_byte_depth) mod 4;
  //Application.MessageBox(PChar(Format('size - %d, color used - %d, size image - %d',[i,j,k])),PChar('Bitmap Offset'),MB_OK);
  ClearCountArray;
  // ---------------------------------
  // Count color number (reduce color)
  for j:=0 to (h1-1) do
```

```
begin
 for i:=0 to (w1-1) do
 begin
  k := image_offset + j*(pixel_byte_depth*w1+padding) + i*pixel_byte_depth;
  if k < size_image then
  begin
   r3 := P2[k+2];
   g3 := P2[k+1];
   b3 := P2[k];
   r1 := r3;
   g1 := g3;
   b1 := b3;
   r2 := r1 div KU_COLOR_CHANEL_STEP;
   g2 := g1 div KU_COLOR_CHANEL_STEP;
   b2 := b1 div KU_COLOR_CHANEL_STEP;
   Inc(color_list[r2,g2,b2].count)
  end
 end
end;
total_valied_color := 0;
for i:=0 to KU_COLOR_CHANEL_RESOLUTION do
for j:=0 to KU_COLOR_CHANEL_RESOLUTION do
for k:=0 to KU_COLOR_CHANEL_RESOLUTION do
begin
 if color_list[i,j,k].count > 0 then
 begin
  Inc (total_valied_color);
 end
end;
ImagePickedColorNumber := total_valied_color;
{
S := '------------------------';
KUColorList1.Items.Add(S);
S := format ('total - %d color(s) avairable', [total_valied_color]);
KUColorList1.Items.Add(S);
}
GlobalUnlock(ImageHandle);

//--------------------------------
// Count flood fill region
```

```
if IslandImageHandle <> 0 then
begin
 IK5FreeMemory (IslandImageHandle);
end;
i := GlobalSize(ImageHandle);
IslandImageHandle := GlobalAlloc (GMEM_FIXED, i);
if IslandIMageHandle = 0 then Exit;
P1 := GlobalLock (IslandImageHandle);
P2 := GlobalLock (ImageHandle);
P3 := Addr(P1[40]);
CopyMemory(P1,P2,40);
FillMemory(P3,i-40,255);
for j:=0 to (h1-1) do
begin
 for i:=0 to (w1-1) do
 begin
  k := image_offset + j*(pixel_byte_depth*w1+padding) + i*pixel_byte_depth;
  island_count := 0;
  if k < size_image then
  begin
   r3 := P2[k+2];
   g3 := P2[k+1];
   b3 := P2[k];
   r1 := r3;
   g1 := g3;
   b1 := b3;
   r3 := P1[k+2];
   //g3 := P1[k+1];
   //b3 := P1[k];
   r4 := r3;
   //g4 := g3;
   //b4 := b3;
   if r4 = 0 then continue;
   r2 := r1 div KU_COLOR_CHANEL_STEP;
   g2 := g1 div KU_COLOR_CHANEL_STEP;
   b2 := b1 div KU_COLOR_CHANEL_STEP;
   check_around_pixel; // check around pixel
   if island_count > 0 then add_island_record
  end
 end
```

```
    end;
  GlobalUnlock (ImageHandle);
  GlobalUnlock (IslandImageHandle);
end;
```

A-2 Sample Pascal code for searching image by color data index

```
cindex := Length(src_rec^.pixel_order)-1;
if cindex > max_color then cindex := max_color; // upper limit
rec := files_head;
while rec<>nil do
begin
 if rec = src_rec then
 begin
  rec := rec^.next;
  continue
 end;
 cindex2 := Length(rec^.pixel_order)-1;
 c_count := 0;
 if cindex2 > max_color then cindex2 := max_color; // upper limit
 for i:=0 to cindex2 do
 begin
  for j:=0 to cindex do
  begin
   color1 := src_rec^.pixel_order[i]^.color;
   color2 := rec^.pixel_order[j]^.color;
   if color1 = color2 then
   begin
    ratio1 := src_rec^.pixel_order[i]^.ratio_pixel;
    ratio2 := rec^.pixel_order[j]^.ratio_pixel;
    if (ratio1>=level_threshould) and (ratio2>=level_threshould) then Inc(c_count)
   end
  end;
 end;
 if c_count >= count_threshould then
 begin
  // save image
  gSearchResult[gNumberSearchResult] := rec;
  Inc(gNumberSearchResult)
 end;
 rec := rec^.next
end;
```

画素の色情報を用いたデジタル画像の評価値と画像検索への応用

畑 田 明 信

概要

　現在，インターネット上で利用されている画像検索は，画像そのものではなく画像に関連づけられたテキストのキーワードを用いて画像の検索を行っている。検索のキーとして画像そのものを利用した画像検索はあまり例を見ない。本稿では，画像を用いた画像検索の試みの一端として，画素の色データを元にした評価値の作成と検索への応用について報告を行う。

　手法としては，デジタルスチルカメラを用いて撮影された画像データを元に，画像に含まれる色を検索し，どの色が何ピクセル存在するか，あるいは，どの色の島がいくつあるのかを集計する。ただし，色の数は 1,600 万色以上存在するが，これほどの数の色を個別に集計することは，評価値の作成には向かないので，RGB 各チャネルつき，256 段階あった色情報を 8 段階程度に減らしてから集計を行った。次に，特定の画像を同一の色や島が上位 X 色程度以内に含まれている画像を画像プールの中から検索を行った。この時，同一の色が Y 個見つかったとすると，それが検索限界値として予め設定した値 T よりも大きいとき，検索がヒットしたと判定する。

　この方法で検索を行ってみると，たしかに，似たイメージに見える画像を検索することができるが，まったく似ていない画像も検索結果として報告される。この手法では，同一の画像を検索させるのではなく，似た画像を検索することになるため，この結果はある程度の予想の範囲である。しかし，より人間が見て似ていると感じられる画像だけを検索結果として取り出すためには，色データの評価についてさらに検討が必要であると思われる。

　今回の研究で行った画像検索のアルゴリズムは，風景画像の評価を行うための手法を応用した物である。この手法による評価値の作成と検索では，画像に含まれる個別の物体や人物を識別して検索することはできない。例えば，特定の人物が写っている画像を探すというような用途には応用することができない。これは，相貌や物体の認知について人間の脳が実現しているメカニズムがその目的に固有の部位と機能に依存しているので，それらに似たアルゴリズムを用意しなければ相貌検索などを実現することは難しい物と思われ，そのようなアルゴリズムは本稿で述べたものとは全く異なったものであると予想される。よって，これは本稿で報告した手法のある種の限界であるといえるだろう。

　一方，まるで似ていない画像を検索結果としてピックアップしてしまうという問題は，インデックスデータを元にした評価値のスコアリング（再解釈・再計算した値）や，色そのものの数の減らし方などによって改善できる物と思われる。

　本稿で報告を行った画像評価および検索の手法は，風景画像の検索に限定する限りでは，原画像に似たものを探すことができるため，今後はより高い精度での検索を実現できるよう，さらに研究を行っていきたい。