第7章 応用処理

吉田 大海

■画像処理の実行プログラム
■画像処理のサンプル画像
https://www.cqpub.co.jp/interface/download/contents.htm

■スマートフォンで体験（7-1から7-14に掲載したQRコードからたどれる図2をまとめたもの）
Wi-Fi接続環境での読み込みをお勧めします．
100Mバイトほどあるので時間が掛かります．

図2はコチラから

7-1 色数を減らして非写実的な立体画像「ポスタリゼーション」

プログラム名：7-1.cpp

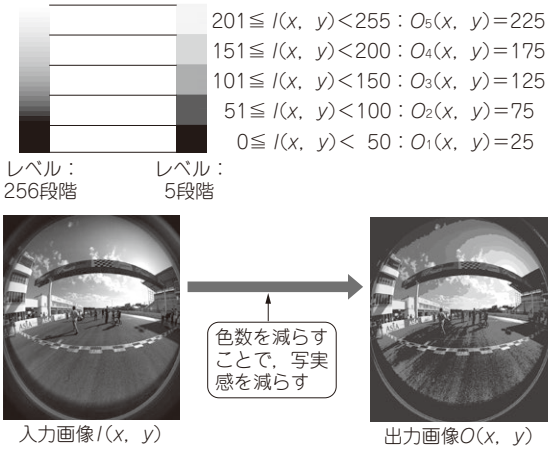

図1 ポスタリゼーション…各チャネルの量子化数を減らす

実写画像と手描き画像の違いは何でしょうか．大きな要素を1つ挙げるなら，色のバリエーションがあります．色ヒストグラムで確認するとよく分かりますが，撮影写真によって得られた画像は色のバリエーションが多く，手描きのポスタでは比較的少ない限られた色数で作成されます．本処理では色数を減らすことで，ポスタ風に画像を変更する処理を紹介します．

● 仕組み

カラー写真やモノクロ写真の色数は，1チャネル当たり8ビットで表現するのが一般的です．つまり256種類あります．カラー画像の場合，R，G，Bの3チャネルがあり，色の組み合わせは$256^3 = 16{,}777{,}216$となります．

このようにチャネルが表現できる色数を量子化数と呼びます．ポスタリゼーションとは，この量子化数を減らす処理です．どのように減らすかは，**図1**の通りで，量子化の一定区間をその中央値に置き換えることで実現します．具体的には量子化の区間t_1からt_2の間の色を，全て$(t_1 + t_2)/2$に置き換えるという操作です．

リスト1 ポスタリゼーションのプログラム（抜粋）

```
for (y = 0; y < Y; y++) {
  for (x = 0; x < X; x++) {
    //(1/)----< 入力画像から画素値を読み込む >----
    for (i = 0; i <= 2; i++) {//BGRチャネルに適用
      p[i] = img.at<cv::Vec3b>(y, x)[i];

      for (l = 0; l < q; l++) {
        t1 = cvRound(l * ((256.0) / (double(q))));
        t2 = cvRound((l + 1)
             * ((256.0) / double((q))));
        P = (t1 + t2) / 2;
        if (t1 <= p[i] && p[i] < t2) { p[i] = P; }
        if (l == (q - 1)) {//最後のループなら
          if (t1 <= p[i]) {
            //t1以上ならポスタリゼーションの値に
            p[i] = P;
          }
        }
      }
      //(3/)/----< 出力画像を入れ込む >----------
      img.at<cv::Vec3b>(y, x)[i] = p[i];
    }
  }
}
```

● 実行結果（ステレオ画像に対する効果）

ポスタリゼーションのプログラムを**リスト1**に示します．**図2**に入力画像とポスタリゼーションを適用した結果を示します．色数を減らされたステレオ画像は写実感が減り，ポスタのような濃淡になっています．この画像を立体視するとポスタの世界をのぞいているようなポスタ風立体画像になるでしょう．

ポスタリゼーションでは量子化数を設定しますが，微妙な濃淡変化は消失しつつも主要な輪郭が損なわれない値を選ぶことが重要になります．この条件を満たせば，立体視した時に立体感が損なわれることはなく，また，ポスタらしさを楽しむことができるでしょう．