第4章 ラズベリー・パイ2＆定番ルックアップ・テーブル方式でサクサク！

画像の高速補正＆投影

鎌田 智也

第3章で求めた平面射影行列を使って，投影ひずみ補正された画像を出力する手順を説明します．いくつかの高速化についても挑戦してみます．

投影プログラムの作成

● OpenCVを使えば簡単1文で変換！

平面射影行列の計算には，長い説明を要しましたが，画像の補正変換処理そのものは簡潔なものです．

OpenCVには，平面射影行列と画像を与えると射影変換を行ってくれる便利な関数が用意されています．

```
void cvWarpPerspective( const CvArr* src, CvArr* dst, const CvMat* mapMatrix, int flags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS,CvScalar fillval=cvScalarAll(0) )
```

あるいは，デフォルト引き数値を省略して次の形式でも大丈夫です．

```
void cvWarpPerspective( const CvArr* src, CvArr* dst, const CvMat* mapMatrix, int flags,CvScalar fillval)
```

引き数のsrcはプロジェクタに出力したいひずみのない画像，dstはコンソール・フレーム・バッファに出力するひずみ補正後の画像，Hはステップ1で求めた平面射影行列です．flagsには，補間方法を指定します．

平面射影行列を使った座標変換処理では，途中の計算で参照すべきピクセル座標の値が小数値（サブピクセル座標値）になり得ます．

しかし画素情報は，整数のピクセル座標位置単位にしかありません．そこでcvWarpPerspectiveには，サブピクセル座標位置の輝度を計算するための補間のしくみがあります．**表1**は，cvWarpPerspectiveで選択できる補間方式です．

左側に示した関数をflagsに指定すれば，所望の補間方式にて画像の変換処理をしてくれます．

表の下にあるものほど，補間に使用する周辺の参照ピクセル数が多いなど処理内容が複雑である反面，変換後の画像にギザが目立たたない傾向となります．下にいくほどCPUの計算負荷は重くなります．

ラズベリー・パイで実行するなら，INTER_NNまたはINTER_LINEARがベストな選択といえます．

各補間手法における処理時間をAppendix3で比較していますので参照ください．

● プログラムの記述

リスト1は，与えられている補正パラメータから平面射影行列を計算し，オリジナル画像をひずみ補正処理して補正後の画像を生成するプログラムです．

リスト2は，ひずみ補正処理した画像をSDLを通じて画面出力するプログラムです．表示するべき画像を更新するたびにこれらの処理を繰り返します．

図1は，CQ出版社のロゴを投影した例です．**図1**（a）のロゴ原画を，あるプロジェクタの取り付け位置と角度でそのまま投影したのが**図1**（b）です．縦方向に伸びて楕円に投影されています．

図1（c）は取り付けを考慮してひずみ補正処理した画像です．**図1**（b）は投影せずに見ると不自然にひずんだ画像ですが実際にプロジェクタで投影すると**図1**

表1　射影変換関数cvWarpPerspectiveでは整数ピクセル座標にしかない輝度の小数ピクセル座標への補間方式を選べる

関　数		名　称	処理内容	処理の重さ
C言語	C++			
CV_INTER_NN	INTER_NEAREST	最近傍法	一番近いピクセルの輝度を使う	軽い
CV_INTER_LINEAR	INTER_LINEAR	線形補間法	周囲4ピクセルで補間	軽い
CV_INTER_CUBIC	INTER_CUBIC	bicubic補間法	周囲16ピクセルで補間	少し重い
CV_INTER_AREA	INTER_AREA	area-based法	モアレ軽減を考慮した補間	軽い
CV_INTER_LANCZOS4	INTER_LANCZOS4	lanczos法	64個（8×8）の近傍ピクセルで補間	一番重い