# Camera Matrix Decomposition

## Standard camera model

従って、rotation matrix が得られたら、以下のようにして回転角、、を求めることができる。

## Our camera model

Rotation around viewing direction (i.e., Z axis of camera coordinates) instead of Z axis of world coordinates.

従って、rotation matrix が得られたら、以下のようにして回転角、、を求めることができる。

# Three vanishing points method

Camera calibration using two or three vanishing points” by Orghidan et al. [2012]に基づいて実装したが、もとの論文は“Camera calibration from vanishing points in images of architectural scenes” by Cipolla et al. [1999]だ。

ただし、OpenGLではright-handed systemで、projection matrixでZ軸を反転させるため、少し修正が必要。

3つのvanishing point、、、が与えられた時、

まず、projection matrixは、上で述べた通り、Z軸を反転させるため、

|  |  |
| --- | --- |
|  | (12) |

次に、世界座標系のX、Y、Z軸方向は、image plane上にprojectすると、image planeの中心から対応するvanishing pointへのベクトルと同じ方向となる。これを式で表すと、

|  |  |
| --- | --- |
|  | (15) |

ここで、は水平方向の左側のvanishing point、は右側のvanishing point、は垂直方向のvanishing pointで、それぞれZ軸、X軸、Y軸に対応していることが分かる。ただし、だけ、Z軸の負の方向を向いているため、マイナスしている。

式(15)より、

|  |  |
| --- | --- |
|  | (16) |

さて、回転行列の各列は互いにorthogonalなので、内積=0。よって、

より、

よって、

|  |  |
| --- | --- |
|  | (18) |

となり、focal lengthを求めることができる。

続いて、式(15)の左辺を分解すると、

|  |  |
| --- | --- |
|  | (19) |

両辺に、それぞれの転置を掛けると、

|  |  |
| --- | --- |
|  | (20) |

左辺を計算すると、

見ると分かる通り、左辺は対象行列だ。この式から、6つの式が得られる。

ここで、右辺に0があるので、たぶん、一発でSVDでは解けないと思う。一番最後の式より、

これを他の5つの式に代入すると、

これで右辺に0がなくなったので、SVDで解ける。

|  |  |
| --- | --- |
|  | (22) |

、が求まったら、も求まる。後は、式(16)に代入して、を求めることが出来る。

# My Approach

基本的に3 vanishing pointsをベースにするが、カメラ中心のimage planeへのprojected point の座標がでないケースも考慮して、focal lengthを計算する。

まず、3つのvanishing pointから、回転行列が得られる。

ここで、、、は、各列をユニットベクトルにするための係数だ。回転行列なので、列ベクトル同士の内積は0。従って、以下の3つの式が得られる。

ゆえに、

ゆえに、

式を整理すると、

行列で表すと、

よって、以下のように、image plane上でのカメラ中心の射影点の座標が求められる。

後は、最初の式に、を代入してやれば、回転行列が計算できる。

なお、は、カメラのprojection matrixで以下のように反映する必要がある。

なぜかと言うと、一般的なprojection matrixは、

ただし、、、、は、image planeの左右上下の座標を表し、を使って以下のように表せる。

よって、上のprojection matrixに代入すると、最初に挙げたprojection matrixが得られる。