進捗報告

B4 田川 幸汰

1 課題 1. パノラマ画像生成

1.1 課題 1 の実装方法

以下にパノラマ画像を生成する方法をまとめる。詳しい理論については以降の節で説明する。[1]

- 入力... 画像 2 つ、対応点データ
- 出力... パノラマ画像
- アルゴリズム
  1. 第一画像、第二画像の対応点データをもとに射影変換行列を求める
  2. 第一画像、第二画像を読み込む
  3. パノラマ画像のサイズを決定する
  4. 第一画像データをパノラマ画像にコピーする
  5. 第二画像データをパノラマ画像にコピーする

1.2 射影変換

パノラマ画像生成に用いる射影変換は一般に次の式で与えられる。変換前の座標を (x, y)、変換後の座標を (x', y') とする。[2]

$$x' = \frac{Ax + Bx + C}{Px + Qx + R} \tag{1}$$

$$y' = \frac{Dx + Ex + F}{Px + Qx + R} \tag{2}$$

射影変換の中でもこの式を用いるものはホモグラフィ変換という。アフィン変換は平行四辺形の変形が可能だが、ホモグラフィ変換では任意の凸四角形に変換することができる。

ここで、R=1 として暫定的に正規化し、式 1、式 2 の分母を払うと、次のような連立方程式を導出することができる。

$$Ax + By + C - Pxx' - Qyx' - x' = 0 \tag{3}$$

$$Dx + Ey + F - Pxy' - Qyy' - y' = 0 \tag{4}$$

式 3、式 4 を $Ah = b$ という行列で表し、最小二乗法を用いて以下の目的関数 J の値を最小にする係数ベクトル h の値を求める。なお、変換前後の座標が四点の場合は、連立方程式の解をそのまま求めればいい。五点以上の場合連立方程式が解けないので最小二乗法を用いる。

$$J = ||Ah - b||^2 \tag{5}$$

J をベクトル h について偏微分することで、パラメータベクトルは次のように求められる。

$$h = (A^T A)^{-1} A^T b \tag{6}$$

1.3 パノラマ画像のサイズの決定

射影変換行列の逆行列 H^{-1} を求めて、第二画像の四隅の座標 $(0,0)$、$(width,0)$、$(0,height)$、$(width,height)$ が第一画像上のどの座標に変換されるか計算する。そして、図 1 のように第一画像の四隅と、変換された第二画像の四隅の点を内包する矩形領域をパノラマ画像のサイズとする。また、オフセットの設定をする。オフセットは、矩形領域の (x_{min},y_{min}) にあたる。

1.4 パノラマ画像の出力

図 2 のように、入力画像 1 の座標をオフセットの分だけ引いて、それに対応する出力画像の座標に入力 1 の画素をコピーする。そして、出力画像の座標にオフセットの分だけ加えて射影変換し、射影変換後の座標が入力 2 の画像に含まれていれば入力 2 の画素をコピーする。

1.5 実行結果

1.5.1 使用する画像

(1) 入力画像 1　　(2) 入力画像 2

図 1: 入力画像

1.5.2 対応点データ

$(386,382)$、$(587,410)$、$(606,244)$、$(516,35)$ → $(55,412)$、$(256,380)$、$(245,225)$、$(126,37)$ 顎の横*2、右側の目の端、額の中心の 4 点を対応点として取得

1.5.3 実行結果

図 1 の二枚の画像から、図 2 のようなパノラマ画像が出力できている。また、オフセットの分だけ上端がずれていることがわかる。python の画像処理系のライブラリである OpenCV の、WarpPerspective メソッドを用いた場合 [3] とそうでない場合を比較すると、実行時間はかなり大きな差が出た。また第二画像を変換して貼り付けた領域は、画像が荒くなっている。これは射影変換で座標を求める際に丸め込みを行ってしまい、正確な画素値が得られていないことが問題だと思われる。

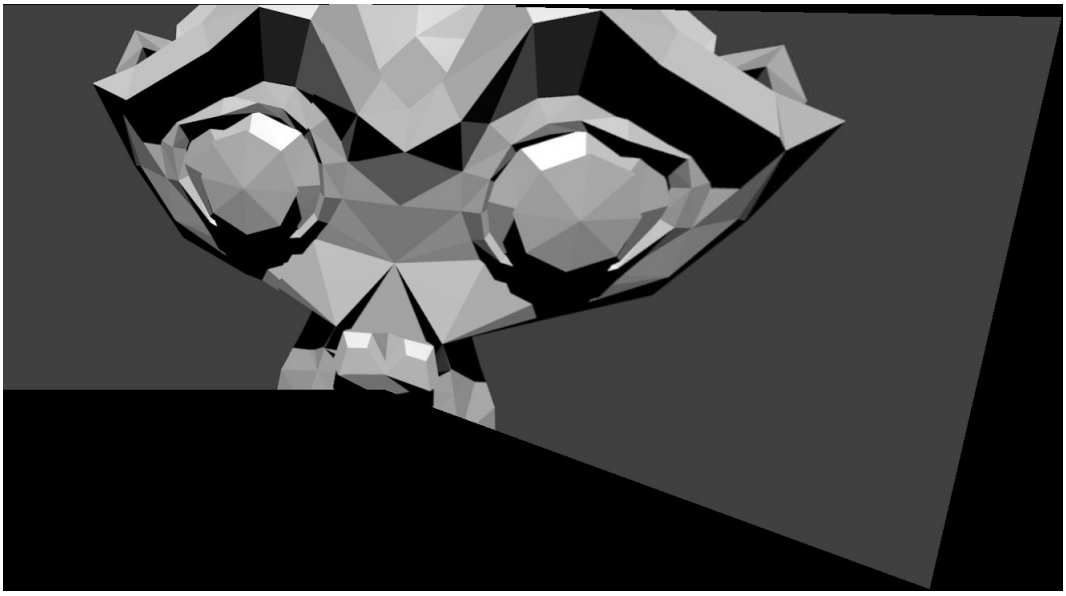

図 2: パノラマ画像

1.6 双一次補完

射影変換を行いパノラマ画像を表示すると、射影変換をした部分の画像が荒く表示される。これは、射影変換で画素座標を整数値に丸めていることが原因である。この問題を解決するため、双一次補完による画素値の内挿を行う。(x, y) を画素座標の整数値、(a, b) を小数部分の値とし、座標 (x, y) に対応する画素値を $I(x, y)$ とすると、双一次補完によって求められる画素値は以下のように計算できる。

$$\begin{aligned} I(x+a, y+b) = (1-b)((1-a)I(x, y) + aI(x+1, y)) \\ +b((1-a)I(x, y+1) + aI(x+1, y+1)) \end{aligned} \tag{7}$$

1.6.1 実行結果

出力画像拡大比較 (右が双一次補完なし、左が双一次補完あり) 図 3 から、双一次補完を行うかどうかで、かなり画像

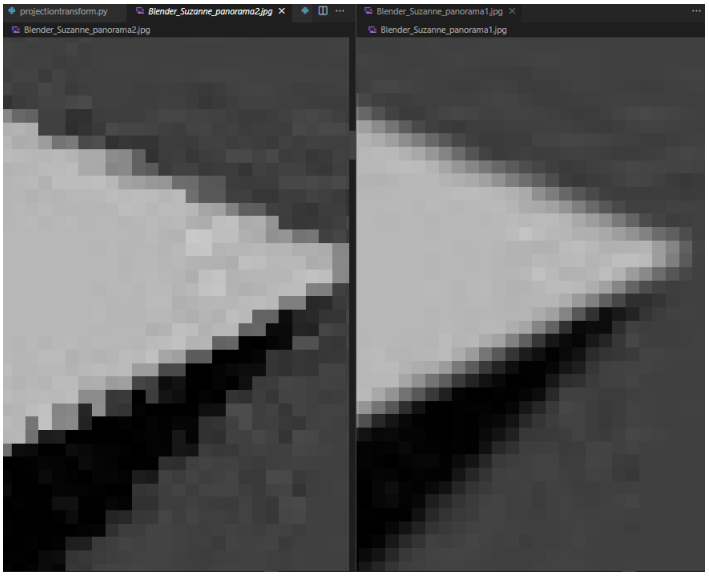

図 3: 双一次補完の有無の比較

の色のつなぎ目の粗さが変わっていることがわかる。

2 課題 2. 全方位画像から透視投影画像生成

2.1 課題 2 の実装方法

以下に全方位画像から透視投影画像を生成する方法をまとめる。詳しい理論については以降の節で説明する。

- 入力... 正距円筒画像 I_e、水平方向及び垂直方向の画角 Θ, Φ、出力画像の画像サイズ H_p, W_p
- 出力... 透視投影画像 I_p
- アルゴリズム
  1. 水平方向、垂直方向の画素間の長さ $\Delta\ x, \Delta\ y$ を計算する。
  2. 透視投影画像 I_p の画素 (u_p, v_p) から、3 次元の視線ベクトルを計算する。
  3. 2 で求めた視線ベクトルから、(θ, ϕ) を計算する。
  4. 距円筒画像 I_e 上の対応する画素 (u_e, v_e) を計算して、その画素値を透視投影画像 I_p の画素 (u_p, v_p) に割り当てる。

2.2 画素間の長さ

水平、垂直方向の画角 Θ, Φ と画像サイズ W_p, H_p が与えられているとき、画像面までの距離を $1(z = 1)$ とすると、水平方向、垂直方向の画素間の長さ $\Delta\ x, \Delta\ y$ は以下のように表せる。

$$\Delta\ x = 2tan(\Theta/2)/W_p \tag{8}$$

$$\Delta\ y = 2tan(\Phi/2)/H_p \tag{9}$$

2.3 視線ベクトルの計算

透視投影画像の画素 (u_p, v_p) とすると、式 9、式 9 で求めた $\Delta\ x, \Delta\ y$ を用いて 3 次元の視線ベクトル $\bar{x}$ は以下のように計算される。

$$\bar{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (u_p - W_p/2)\ \Delta\ x \\ (v_p - H_p/2)\ \Delta\ y \\ 1 \end{pmatrix} \tag{10}$$

2.4 透視投影画像への投影

正距円筒画像の一部を透視画像の画像面に投影することを考える。正距円筒画像は球面上に投影した画像とも考えられるので、透視投影の画像面の画素に対応する正距円筒画像の角度情報が得られれば、正距円筒画像の画素を透視投影画像に投影できる 。透視投影面上の座標 (x, y, z) から、角度 (θ, ϕ) は次のように計算できる

$$\theta = tan^{-1}\frac{x}{z} \tag{11}$$

$$\phi = -tan^{-1}\frac{y}{\sqrt{x^2 + z^2}} \tag{12}$$

ただし、X 軸周りの回転は Y 軸から Z 軸方向への回転が正の回転であることを考えて、ϕ の符号は反転している。

2.5 360 度画像の座標系変換

360 度カメラで撮影した画像は、正距円筒画像として保存される。これはシーンを球面に投影したものを緯度、軽度を画像の縦方向と横方向に対応付けて矩形の画像として再現したものである。
カメラ画像の z 軸 (光軸) が正距円筒画像の中心を通るとき、画像中心の θ-ϕ 座標系での座標が $(\theta, \phi) = (0, 0)$ となると考えると、画像の左上を原点として、右方向に u 軸、下方向に v 軸をとる座標系との関係は以下のように表せる。

$$(\theta, \phi) = ((u - \frac{W}{2})\frac{2\pi}{W}, (\frac{H}{2} - v)\frac{\pi}{H}) \tag{13}$$

$$(u, v) = ((\theta + \pi)\frac{W}{2\pi}, (\frac{\pi}{2} - \phi)\frac{H}{\pi}) \tag{14}$$

ここで、W 及び H は入力画像の幅及び高さである。

2.6 実行結果

2.6.1 使用する画像

図 4: 全方位画像

2.6.2 出力画像の画素、サイズ

画角 $(\Theta, \Phi) = (45, 45)$、画像サイズ $(W_p, H_p) = (500, 500)$

2.6.3 実行結果

画質は荒いが透視投影画像が出力できている。視線移動については実装中である。

3 今後の計画

- 課題 2. 透視投影画像の視線移動の完成 (来週まで)
- 課題 3 の実施 (来週まで)

図 5: 透視投影画像

参考文献

[1] 菅谷 保之, “TEO ライブラリによる画像処理プログラミング入門”, 研究室内資料, 2006.

[2] 菅谷 保之, “射影変換行列の計算”, 研究室内資料, 最終更新日 2021/04/07.

[3] “複数の画像からパノラマを作成 (OpenCV、Python)”, https://www.qoosky.io/techs/dea950ec9a, 閲覧日 2023/04/12.

[4] 菅谷 保之, “360 度画像からの透視投影画像の生成 (3), 研究室内資料, 最終更新日 2023/04/05.