A Homogeneous Transformation Class for the ITK

Release 1.00

Rupert Brooks and Tal Arbel

March 2, 2007

McGill Centre for Intelligent Machines
McGill University, Montreal, Canada
rupert.brooks@mcgill.ca, arbel@cim.mcgill.ca

Abstract

Homogeneous transformations are widely used in computer vision and graphics. They commonly arise when considering the transformation of an image of a planar object under arbitrary camera motion, or the transformation of two images of the same scene due to camera motion holding the optical center fixed. In this paper we describe the addition of a class of homogeneous transforms for the Insight Toolkit.

Contents

1 Introduction

This paper describes a class which performs homogeneous transformations (which are also called projective transformations) in the ITK framework. These are widely used in computer vision and graphics where

perspective transformations are common due to the use of perspective cameras. These transformations are less common but still occasionally used in medical imaging. Briefly, a homogenous transformation of a point is performed by expressing the point in *homogeneous coordinates* and performing a matrix transformation of that point. These transformations form a group, of which the affine, similarity, Euclidean and translation transforms are subgroups. They are described briefly below, detailed description may be found in [3], [4], or [2].

1.1 Homogeneous Coordinates

The homogeneous coordinates of an n-dimensional point are an $n+1$ dimensional vector of the following form $\begin{bmatrix} \lambda\mathbf{p} & \lambda \end{bmatrix}$, where λ is a scale factor, and $\mathbf{p}$ is the (n-dimensional) coordinate of the point. There are many equivalent homogeneous coordinates for a single point depending on the scale factor, λ. We can obtain a valid homogeneous point from an n-dimensional coordinate simply by appending a final 1. Conversely, we can obtain the n-dimensional coordinate from the homogeneous coordinates simply by dividing by the last element.

If the scale factor λ is zero, then this is considered to be a point at infinity which can also be thought of as a direction. This is equivalent to a vector as defined for the `itk::MatrixOffsetTransform` and derived classes.

1.2 Homogeneous transformation

A homogeneous transformation is performed by left multiplying homogeneous coordinates by a square matrix of dimension $n+1$. If the matrix is restricted to be invertible, then this set of transformations forms a group, of which the affine transformations are a subgroup. Although not strictly necessary, it is often convenient to define a center about which the transformation acts. This center is implemented by pretransforming the coordinates so that the center is located at the origin, applying the transformation, and then posttransforming the result by the inverse of the initial, centering transformation.

Specifically, if we parameterize the elements of the matrix using $\{a,b,c,...\}$ as shown below, then the transformation of a point (for the 2D case) will be

$$\begin{bmatrix} \lambda x_{out} \\ \lambda y_{out} \\ \lambda \end{bmatrix} = \begin{bmatrix} 1 & 0 & C_x \\ 0 & 1 & C_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & e \\ c & d & f \\ g & h & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -C_x \\ 0 & 1 & -C_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{in} \\ y_{in} \\ 1 \end{bmatrix} \tag{1}$$

thus the new coordinates of a point are given by:

$$x_{out} = \frac{a(x_{in}-C_x)+b(y_{in}-C_y)+e}{g(x_{in}-C_x)+h(y_{in}-C_y)+1}+C_x \tag{2}$$

$$y_{out} = \frac{c(x_{in}-C_x)+d(y_{in}-C_y)+f}{g(x_{in}-C_x)+h(y_{in}-C_y)+1}+C_x \tag{3}$$

where $\begin{bmatrix} C_x & C_y \end{bmatrix}$ are the coordinates of the center point. Note that these equations are non-linear in general. If the elements in the bottom row of the matrix, g and h are equal to zero, then this transformation reduces to the familiar affine transformation.

2 Proposed Class and Implementation

The class proposed, `itk::HomogeneousTransform`, implements the Homogeneous transformation of n-dimensional points as described above. Internally, the class stores a $n+1 \times n+1$ matrix representing the transformation, and the center about which the transformation acts (an n-dimensional point).

2.1 Parameterization

The transformation is parameterized using one parameter for each element of the matrix, except for the bottom left entry, which is assumed to be one. These are arranged so that the first $n \times n$ entries correspond to the affine matrix part of the transformation, the following n entries correspond to the translation, and the final n entries are the perspective, or *elation* part of the transformation.

This parameterization is fairly widely used (e.g. [1]) and intuitively understandable, but is flawed in that it cannot represent all possible valid homogeneous transformations. As an example, consider that there are invertible matrices, which have a zero element in the lower left corner. These are valid homogeneous transforms, but cannot be represented in this scheme.

This parameterization was used despite this problem, as these transformations occur rarely in practice. They correspond to moving a perspective camera through the object being viewed and out the other side. Furthermore, complete parameterizations suffer from other difficulties. They either are not *minimal*, that is they have more parameters than the inherent dimensionality of the space, or they require matrix exponentiation, making them computationally difficult to evaluate.

2.2 Transformation of points

Transformation of points is implemented in the obvious way, using the following algorithm:

Algorithm 1 Transformation of points

Extend input point to homogeneous point by appending 1
Subtract the center coordinates from input point.
Multiply the result by the transformation matrix.
Add the center coordinates to the transformed point.
Convert the output point to a non-homogeneous point by dividing by the final component.

Back transformation is performed similarly, except that the inverse transformation matrix is used.

Transformation of vectors is not supported. A vector, or direction, in projective space may be considered as a point on the plane at infinity. Homogeneous transformations have the ability to move the plane at infinity. While this remains meaningful while working in homogeneous coordinates, the results do not really make sense when converted back to non-homogenous coordinates.

2.3 Composition

Composition of transformations which have the same center is easily done by multiplying their matrices. To support composition when the centers of the transformations are different, two supporting methods were

implemented. The `DeCenter()` method returns the matrix for an equivalent transform centered at the origin. Conversely, the `ReCenter()` method, returns an equivalent transformation centered on a new center point.

It is important to differentiate the `ReCenter()` method from the `SetCenter()` method. The `SetCenter()` method simply sets the position of the center point. After `ReCenter()`, points will be transformed identically, but the transformation matrix will be different. After `SetCenter()` however, the transformation matrix is identical, but points will be transformed differently (because the center point is different).

Given the `DeCenter()` and `ReCenter()` methods, composition is easy. Both transformations are decentered, multiplied together, and then recentered on the appropriate center point. Composition with the `itk::MatrixOffsetTransform` and derived classes is supported by first computing an equivalent homogeneous transform to the `itk::MatrixOffsetTransform` and then composing with that in the above manner.

2.4 Jacobian Matrix

The Jacobian matrix of the transformation has a reasonably simple form in terms of the input and output homogeneous points. Let $\tilde{x}_i$ refer to the input coordinates after the center is subtracted, indexed by i,

$$\tilde{x}_i = x_{in_i} - C_i, \tag{4}$$

and let $\hat{x}_i$ be the raw homogeneously transformed coordinates, along with the output scale factor λ. Specifically,

$$\begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_n \\ \lambda \end{bmatrix} = H \cdot \begin{bmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_n \\ 1 \end{bmatrix} \tag{5}$$

and,

$$x_{out_i} = \frac{\tilde{x}_i}{\lambda} + C_i \tag{6}$$

Then, given the ordering of the parameters used here, the Jacobian matrix can be divided into three parts:

$$J(\bar{\mathbf{x}}_{in}) = \frac{\partial \bar{\mathbf{x}}_{out}}{\partial \bar{\phi}} = \begin{bmatrix} D \\ E \\ F \end{bmatrix} \tag{7}$$

The block D is the block diagonal matrix of derivatives with respect to the affine part of the matrix.

$$D_{ij} = \begin{cases} \frac{\tilde{x}_i}{\lambda} & \text{if } (j-1)n < i <= jn \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

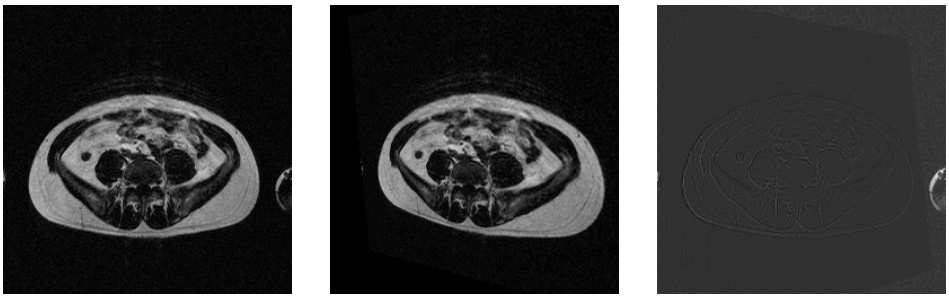
where i ranges from 1 to n^2 and j from 1 to n.

The block E is the $n \times n$ matrix of derivatives with respect to the translation components.

$$E_{ij} = \begin{cases} \frac{1}{\lambda} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Finally the block F is the $n \times n$ matrix of derivatives with respect to the projective components.

$$F_{ij} = -\frac{\tilde{x}_i \hat{x}_j}{\lambda^2} \tag{10}$$



(a) FatMRISlice Original (b) FatMRISlice Warped (c) Difference after registration

Figure 1: Example registration of a 2D image slice

3 Testing

In keeping with the ITK standard, all methods of the class are tested by the testing program `itkHomogeneousTransformTest.cxx`. All tests are performed for both the 2 and 3 dimensional cases. Each case has been hand calculated independently to verify the output of the class.

Furthermore, as this class is intended to be used in the registration framework, we present the results of an example registration program, `HomogeneousRegistration.cxx` based on the ITK example `ImageRegistration3.cxx` in Figure 1. This example uses the **FatMRISlice.png** image from the ITK example data. The image has been warped by a small homogeneous transform, which is successfully resolved by the registration process. It should be noted, however, that scaling is important for this transform. The final two parameters of the transform should be given particularly large scales.

4 Conclusion

This paper presented the class `itk::HomogeneousTransform` which performs homogeneous, or projective, transformations of n-D coordinates. The transformation is designed to fit into the ITK registration framework; in particular, the Jacobian matrix is available, so it can be used by gradient based optimizers. A testing program which tests all methods of the class has been provided.

References

[1] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unified framework. *International Journal of Computer Vision*, 56(3):221–255, 2004. 2.1

[2] Jules Bloomenthal and Jon Rokne. Homogeneous coordinates. *The Visual Computer*, 11(1):15–26, 1994. 1

[3] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003. 1

[4] Marc Pollefeys. Visual 3d modeling from images. Online course notes, 2002. Available from `http://www.cs.unc.edu/~marc/teaching.html`. 1