Decomposition of Homogeneous 4×4 Matrices

Rammi* (rammi@caff.de)

April 14, 2020

Abstract

This document gives a step-by-step instruction how to decompose a general homogeneous 4×4 matrix as commonly used in 3D graphics into simpler transformations like rotations, scalings and translations. After the first step the effective matrix is already reduced to an affine 4×3 matrix, and after the second step to 3×3, so later decompositions steps can be used for these, too.

A short overview how to do the same for affine 3×2 matrices as used in 2D graphics is also provided.

Contents

1 Definitions

In the following we'll use column vectors and matrix multiplication from the left for transforming such vectors.

A matrix will be represented by an uppercase bold letter: $\mathbf{M}$.

A vector is denoted by a bold lowercase letter ($\mathbf{v}$), a single coordinate of a vector by a non-bold lowercase letter with one index, and a matrix component by a non-bold lowercase letter two indices. A matrix:

$$\mathbf{A} = \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} & a_{xw} \\ a_{yx} & a_{yy} & a_{yz} & a_{yw} \\ a_{zx} & a_{zy} & a_{zz} & a_{zw} \\ a_{wx} & a_{wy} & a_{wz} & a_{ww} \end{pmatrix}$$

A vector:

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \\ v_w \end{pmatrix}$$

Multiplying both (i. e. transforming the vector):

$$\mathbf{Av} = \begin{pmatrix} a_{xx}v_x + a_{xy}v_y + a_{xz}v_z + a_{xw}v_w \\ a_{yx}v_x + a_{yy}v_y + a_{yz}v_z + a_{yw}v_w \\ a_{zx}v_x + a_{zy}v_y + a_{zz}v_z + a_{zw}v_w \\ a_{wx}v_x + a_{wy}v_y + a_{wz}v_z + a_{ww}v_w \end{pmatrix}$$

This means that when applying a chain of transformations to the same vector the transformations are multiplied in inverse order. I. e. when a vector is first transformed with a rotation $\mathbf{R}$ and then scaled with $\mathbf{S}$ this will result in a combined transformation $\mathbf{M}$

$$\mathbf{M} = \mathbf{SR}$$

It is required that matrices can be transposed

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$$

and inverted

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{1}$$

(introducing also the identity transformation $\mathbf{1}$).

2 4×4 Matrix Decomposition

The overall goal is to reduce any invertable 4×4 matrix $\mathbf{A}$ into 5 simpler transformations:

- a projection transformation $\mathbf{P}$
- a translation $\mathbf{T}$
- a rotation $\mathbf{R}$
- a shearing transformation $\mathbf{H}$
- a scaling transformation $\mathbf{S}$ (which will also include possible mirroring)

so that $\mathbf{A}$ can be written as the product of the above simpler transformations

$$\mathbf{A} = \mathbf{PTRHS}$$

The order used here is chosen to make some operations a bit easier, but other orders are possible and the description of the steps should be detailed enough to allow to derive the necessary steps for other orders.

2.1 Step 1: Extraction of a Projection Transformation

Here the term *projection transformation* is used for a transformation of the form

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p_{wx} & p_{wy} & p_{wz} & p_{ww} \end{pmatrix}$$

A projection is the only transformation in the resulting chain which may turn 3D vectors (i. e. 4D vectors with $w = 0$) into points (i. e. 4D vectors with $w \neq 0$, often normalized to $w = 1$) and vice versa. All other simple transformations leave the w coordinate untouched.

The first decomposition is to extract the projection by calculating

$$\mathbf{A} = \mathbf{PB}$$

with $\mathbf{B}$ being a 4×3 matrix extended to 4×4 (which will be decomposed in the further steps)

$$\mathbf{B} = \begin{pmatrix} b_{xx} & b_{xy} & b_{xz} & b_{xw} \\ b_{yx} & b_{yy} & b_{yz} & b_{yw} \\ b_{zx} & b_{zy} & b_{zz} & b_{zw} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Multiplying results in

$$\mathbf{PB} = \begin{pmatrix} b_{xx} & b_{xy} & b_{xz} & b_{xw} \\ b_{yx} & b_{yy} & b_{yz} & b_{yw} \\ b_{zx} & b_{zy} & b_{zz} & b_{zw} \\ a_{wx} & a_{wy} & a_{wz} & a_{ww} \end{pmatrix}$$

with (we inserted the matched a coefficients above as a shortcut)

$$\begin{aligned} a_{wx} &= p_{wx}b_{xx} + p_{wy}b_{yx} + p_{wz}b_{zx} \\ a_{wy} &= p_{wx}b_{xy} + p_{wy}b_{yy} + p_{wz}b_{zy} \\ a_{xz} &= p_{wx}b_{xz} + p_{wy}b_{yz} + p_{wz}b_{zz} \\ a_{ww} &= p_{wx}b_{xw} + p_{wy}b_{yw} + p_{wz}b_{zw} + p_{ww} \end{aligned}$$

By comparing the coefficients with the incoming $\mathbf{A}$ matrix it is immediately clear that $b_{ij} = a_{ij}$ for all required b coefficients.

The simplest way to calculate $\mathbf{P}$ is the observation that the a_{wj} are the result of the multiplication of the transposed matrix $\mathbf{B}^T$ with a 4D vector

$$\mathbf{p} = \begin{pmatrix} p_{wx} \\ p_{wy} \\ p_{wz} \\ p_{ww} \end{pmatrix}$$

i. e.

$$\begin{pmatrix} a_{wx} \\ a_{wy} \\ a_{wz} \\ a_{ww} \end{pmatrix} = \mathbf{B}^T \mathbf{p}$$

with

$$\mathbf{p} = (\mathbf{B}^T)^{-1} \begin{pmatrix} a_{wx} \\ a_{wy} \\ a_{wz} \\ a_{ww} \end{pmatrix}$$

Inverting a 3×3 matrix is less expensive, so rather create an already transposed matrix from a 3×3 submatrix of $\mathbf{B}$

$$\mathbf{B}' = \begin{pmatrix} a_{xx} & a_{yx} & a_{zx} \\ a_{xy} & a_{yy} & a_{zy} \\ a_{xz} & a_{yz} & a_{zz} \end{pmatrix}$$

invert that, multiply with a 3D vector created from the last row of $\mathbf{A}$ to get the first 3 components of $\mathbf{P}$s last row:

$$\mathbf{p}' = \begin{pmatrix} p_{wx} \\ p_{wy} \\ p_{wz} \end{pmatrix} = \mathbf{B}'^{-1} \begin{pmatrix} a_{wx} \\ a_{wy} \\ a_{wz} \end{pmatrix}$$

The last missing component of $\mathbf{P}$ is then calculated as

$$p_{ww} = a_{ww} - \mathbf{p}' \odot \begin{pmatrix} a_{xw} \\ a_{yw} \\ a_{zw} \end{pmatrix}$$

(with $\odot$ denoting the dot product).

2.2 Step 2: Extraction of a Translation Transformation

In this subsection we operate on the 4×3 matrix $\mathbf{B}$ which is now decomposed into a translation and a matrix $\mathbf{C}$ which is effectively a 3×3 matrix.

$$\mathbf{B} = \mathbf{TC}$$

The solution is simple:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & b_{xw} \\ 0 & 1 & 0 & b_{yw} \\ 0 & 0 & 1 & b_{zw} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$\mathbf{C} = \begin{pmatrix} b_{xx} & b_{xy} & b_{xz} & 0 \\ b_{yx} & b_{yy} & b_{yz} & 0 \\ b_{zx} & b_{zy} & b_{zz} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Also note that $b_{ij} = a_{ij}$ for all relevant coefficients of $\mathbf{B}$ (i. e. $j \neq w$).

2.3 Step 3: Extraction of a Rotation Transformation

Now we are effectively down to a 3×3 matrix, and in the following 4th row and column are no longer displayed.

For extracting the rotation we assume that we have

$$\mathbf{C} = \mathbf{RD} \tag{1}$$

with $\mathbf{R}$ being the rotation and $\mathbf{D}$ a triangle matrix

$$\mathbf{D} = \begin{pmatrix} d_{xx} & d_{xy} & d_{xz} \\ 0 & d_{yy} & d_{yz} \\ 0 & 0 & d_{zz} \end{pmatrix}$$

Rotations have a special property which is used in the next step:

$$\mathbf{R}^T = \mathbf{R}^{-1} \tag{2}$$

Calculating

$$\begin{aligned} \mathbf{C}^T\mathbf{C} &= (\mathbf{RD})^T(\mathbf{RD}) \\ &= \mathbf{D}^T\mathbf{R}^T\mathbf{RD} \end{aligned}$$

and with the help of equation (2) $\mathbf{R}^T\mathbf{R} = \mathbf{1}$ and therefore

$$\mathbf{C}^T\mathbf{C} = \mathbf{D}^T\mathbf{D}$$

So calculating

$$\mathbf{C}' = \mathbf{C}^T\mathbf{C}$$

and identifying its components with

$$\mathbf{D}^T\mathbf{D} = \begin{pmatrix} d_{xx}^2 & d_{xx}d_{xy} & d_{xx}d_{xz} \\ d_{xx}d_{xy} & d_{xy}^2 + d_{yy}^2 & d_{yz}d_{xz} + d_{yy}d_{yz} \\ d_{xx}d_{xz} & d_{yz}d_{xz} + d_{yy}d_{yz} & d_{xz}^2 + d_{yz}^2 + d_{zz}^2 \end{pmatrix}$$

results in the following

$$\begin{aligned} d_{xx} &= \pm\sqrt{c'_{xx}} \\ d_{xy} &= \frac{c'_{xy}}{d_{xx}} \\ d_{xz} &= \frac{c'_{xz}}{d_{xx}} \\ d_{yy} &= \pm\sqrt{c'_{yy} - d_{xy}^2} \\ d_{yz} &= \frac{c'_{yz} - d_{xy}d_{xz}}{d_{yy}} \\ d_{zz} &= \pm\sqrt{c'_{zz} - d_{xz}^2 - d_{yz}^2} \end{aligned}$$

The $\pm$ give us a bit of freedom. We want the rotation to be pure, so any mirroring should go into $\mathbf{D}$. Therefore if $\det(\mathrm{C}) > 0$ we always choose $+$, and if $\det(\mathrm{C}) < 0$ we always choose $-$ (just out of symmetry). This efficiently moves any mirroring into $\mathbf{D}$.

Note that the determinant cannot be 0 because we already inverted $\mathbf{C}^T$ above.

$\mathbf{R}$ can now be calculated (compare (1)) via

$$\mathbf{R} = \mathbf{C}\mathbf{D}^{-1}$$

Basically a rotation can be decomposed into simpler rotations, but this is not handled here, as any rotation can be defined by just an axis and an angle.

2.4 Step 3: Decompositon of the Remaining Triangle Matrix into Shearing and Scaling

The last step is to divide **D** into a shearing

$$\mathbf{H} = \begin{pmatrix} 1 & h_{xy} & h_{xz} \\ 0 & 1 & h_{yz} \\ 0 & 0 & 1 \end{pmatrix}$$

and a scaling transformation

$$\mathbf{S} = \begin{pmatrix} s_{xx} & 0 & 0 \\ 0 & s_{yy} & 0 \\ 0 & 0 & s_{zz} \end{pmatrix}$$

so that

$$\begin{aligned} \mathbf{D} &= \mathbf{HS} \\ &= \begin{pmatrix} s_{xx} & s_{xx}h_{xy} & s_{xx}h_{xz} \\ 0 & s_{yy} & s_{yy}h_{yz} \\ 0 & 0 & s_{zz} \end{pmatrix} \end{aligned}$$

Comparing coefficients results in

$$\mathbf{H} = \begin{pmatrix} 1 & d_{xy}/d_{xx} & d_{xz}/d_{xx} \\ 0 & 1 & d_{yz}/d_{yy} \\ 0 & 0 & 1 \end{pmatrix}$$

and

$$\mathbf{S} = \begin{pmatrix} d_{xx} & 0 & 0 \\ 0 & d_{yy} & 0 \\ 0 & 0 & d_{zz} \end{pmatrix}$$

3 3×2 Matrix Decomposition

3×2 matrices are the standard matrices used in 2D graphics. Your main problem applying the following is to find out if your system uses column vectors (like we do here) or row vectors.

I'm not aware of any 2D graphics which works with projection, therefore we are not starting with 3×3 matrices and don't extract a projection. So the resulting chain of transformations resulting here will be

- a translation $\mathbf{T}$
- a rotation $\mathbf{R}$
- a shearing transformation $\mathbf{H}$
- a scaling transformation $\mathbf{S}$ (which will also include possible mirroring)

or, when $\mathbf{B}$[1] is the matrix to be decomposed

$$\mathbf{B} = \mathbf{TRHS}$$

and again, when thinking of the order how they are applied to a point or vector, it's the inverse of the above.

As this problem is the simpler sister of the 4×4 decomposition the pace is a bit faster.

Starting point is the matrix

$$\mathbf{B} = \begin{pmatrix} b_{xx} & b_{xy} & b_{xw} \\ b_{yx} & b_{yy} & b_{yw} \end{pmatrix}$$

3.1 Step 1: Extraction a Translation Transformation

Simple:

$$\begin{aligned} \mathbf{B} &= \mathbf{TC} \\ \mathbf{T} &= \begin{pmatrix} 1 & 0 & b_{xw} \\ 0 & 1 & b_{yw} \end{pmatrix} \\ \mathbf{C} &= \begin{pmatrix} b_{xx} & b_{xy} & 0 \\ b_{yx} & b_{yy} & 0 \end{pmatrix} \end{aligned}$$

3.2 Step 2: Extraction of a Rotation Transformation

Starting point is $\mathbf{C}$ from above. Goal is

$$\mathbf{C} = \mathbf{RD}$$

[1] Starting with $\mathbf{B}$ here makes it a lot less confusing to compare with the 4×4 section where there is an additional first step.

with a rotation $\mathbf{R}$ and a triangle matrix $\mathbf{D}$

$$\mathbf{D} = \begin{pmatrix} d_{xx} & d_{xy} \\ 0 & d_{yy} \end{pmatrix}$$

Again we'll use eq. (2) to identify

$$\mathbf{C}' = \mathbf{C}^T\mathbf{C} = \mathbf{D}^T\mathbf{D}$$

Calculate $\mathbf{C}'$. Then

$$\begin{aligned} d_{xx} &= \pm\sqrt{c'_{xx}} \\ d_{xy} &= \frac{c'_{xy}}{d_{xx}} \\ d_{yy} &= \pm\sqrt{c'_{yy} - d_{xy}^2} \end{aligned}$$

Once again there is a bit of freedom in choosing the sign of the square roots, to guarantee that $\mathbf{R}$ a pure rotation without mirroring make both + if $\det(\mathbf{C}) > 0$ and one of it $-$ otherwise. Finally

$$\mathbf{R} = \mathbf{C}\mathbf{D}^{-1}$$

3.3 Step 3: Decompostion of the Remaining Triangle Matrix into Shearing and Scaling

The last step is to divide $\mathbf{D}$ into a shearing

$$\mathbf{H} = \begin{pmatrix} 1 & h_{xy} \\ 0 & 1 \end{pmatrix}$$

and a scaling transformation

$$\mathbf{S} = \begin{pmatrix} s_{xx} & 0 \\ 0 & s_{yy} \end{pmatrix}$$

so that

$$\mathbf{D} = \mathbf{H}\mathbf{S}$$

This results in

$$\begin{aligned} s_{xx} &= d_{xx} \\ s_{yy} &= d_{yy} \\ h_{xy} &= \frac{d_{xy}}{s_{yy}} \end{aligned}$$

A Web Resources

- A possibly updated copy of this document can be found under `https://caff.de/posts/4X4-matrix-decomposition/decomposition.pdf`.
- Example source code is found under `https://caff.de/posts/4X4-matrix-decomposition/`.

B Adaptions

B.1 Row Vectors

As a row vector is just a transposed column vector mostly transposing the matrices should do the job. In case of extracting the rotation take care not to be confused because of the additional transposition.

B.2 Other Orders

Extracting the projection and the translation in the first two steps comes quite natural as you'll loose a dimension each time. So I'd definitely stick to that.

Swapping the order of $\mathbf{R}$ and the triangle matrix $\mathbf{D}$ in step 3 in section 2.3 can easily be done. Obviously this also swaps the transposition, so $\mathbf{C}'$ becomes

$$\mathbf{C}' = \mathbf{C}\mathbf{C}^T = \mathbf{D}\mathbf{D}^T$$

Therefore the coefficients are calculated a bit differently. To find out just perform the rightmost product on paper, identify the resulting coefficients with the corresponding $\mathbf{C}'$ coefficients, and resolve. This is easy and straight forward.

A similar approach works when swapping shearing $\mathbf{H}$ and scaling $\mathbf{S}$.

This allows for 4 out of 6 permutations of $\mathbf{R}$, $\mathbf{H}$, and $\mathbf{S}$. Putting $\mathbf{R}$ between the 2 others is not that simple with the approach used here.