

MGH multiple pinhole projection geometry and 3D DRRs

Bill Worstell

PicoRad->MGH

2/7/2024



Introduction:

- We are interested in learned reconstruction and in SPECT/CT
- We have prepared a 1000-event Pytorch tensor dataset from simulated (analytically forward projected by diffDRR) Derenzo phantom 3-D images with high (256x256x256 @ 1x1x1mm) spatial resolution.
- These images and their projections are from a random set which varies uniformly under translation in 3D across a central part of the field of view, and randomly over 3 Euler rotations for phantom orientation.
- We would like to prepare a learned (end-to-end) reconstruction pipeline and want it to be accurate, quick and inexpensive to train, unbiased and robust against noise and novel data instances.
- The linear algebra of projective geometry and homogeneous coordinates leads one to extend Digital Reconstructed Radiographs (dRRs) from the usual 2D to 3D, with a 1-to-1 correspondence between points in the Euclidean image volume and points in a 3d dRR for each projection module (camera) with the third dimension quantity k scaling with the reciprocal of the distance from pinhole to voxel. There are thus 80x256x256x256 projection tensors all of which we can map to and from a known truth source image – viewing this source volume from 80 different perspectives. We can use image maps to transform between.
- We are considering a neural net architecture where the first block is a 3d U-net from a set of 80x256x256 data from each event/run to 80 256x256x256 images, followed by a second block using a 3d U-net from these 80 perspective volume images to a single output 3d Euclidean image.

Projective geometry has lots of developed methods and linear algebra which is documented extensively on Wikipedia for example.

Under projective transformations, linearity is preserved and straight lines map into straight lines.

Projective geometry

Article [Talk](#)

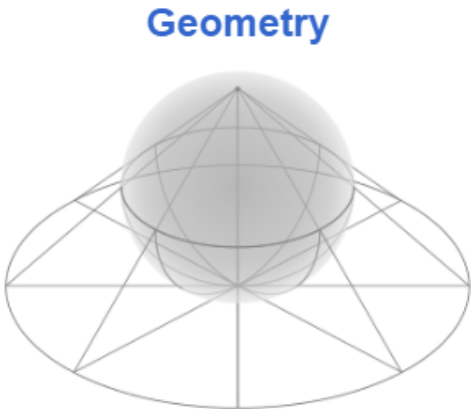
 43 languages 

[Read](#) [Edit](#) [View history](#) [Tools](#) 

From Wikipedia, the free encyclopedia

In [mathematics](#), **projective geometry** is the study of geometric properties that are invariant with respect to [projective transformations](#). This means that, compared to elementary [Euclidean geometry](#), projective geometry has a different setting, [projective space](#), and a selective set of basic geometric concepts. The basic intuitions are that projective space has more points than [Euclidean space](#), for a given dimension, and that [geometric transformations](#) are permitted that transform the extra points (called "[points at infinity](#)") to Euclidean points, and vice versa.

Properties meaningful for projective geometry are respected by this new idea of transformation, which is more radical in its effects than can be expressed by a [transformation matrix](#) and [translations](#) (the [affine transformations](#)). The first issue for geometers is what kind of geometry is adequate for a novel situation. It is not possible to refer to [angles](#) in projective geometry as it is in [Euclidean geometry](#), because angle is an example of a concept not invariant with respect to projective transformations, as is seen in [perspective drawing](#) from a changing perspective. One source for projective geometry was indeed the theory of perspective. Another difference from elementary geometry is the way in which [parallel lines](#) can be said to meet in a [point at infinity](#), once the concept is translated into projective geometry's terms. Again this notion has an intuitive basis, such as railway tracks meeting at the horizon in a perspective drawing. See [Projective plane](#) for the basics of projective geometry in two dimensions.



Projecting a [sphere](#) to a [plane](#)

[Outline](#) · [History](#) ([Timeline](#))

[Branches](#) [hide]

- [Euclidean](#) · [Non-Euclidean](#) ([Elliptic](#) ([Spherical](#)) · [Hyperbolic](#)) · [Non-Archimedean geometry](#) · **[Projective](#)** · [Affine](#) · [Synthetic](#) · [Analytic](#) · [Algebraic](#) ([Arithmetic](#) · [Diophantine](#)) · [Differential](#) ([Riemannian](#) · [Symplectic](#) · [Discrete differential](#)) · [Complex](#) · [Finite](#) · [Discrete/Combinatorial](#) ([Digital](#)) · [Convex](#) ·

Homogeneous coordinates are a natural and useful coordinate system when operating in a projective geometry

They provide simple image-to-image transforms when transforming from Euclidean volumes representations to equivalent projection space coordinates for the same point in 3d space.

Homogeneous coordinates

🌐 20 languages ▼

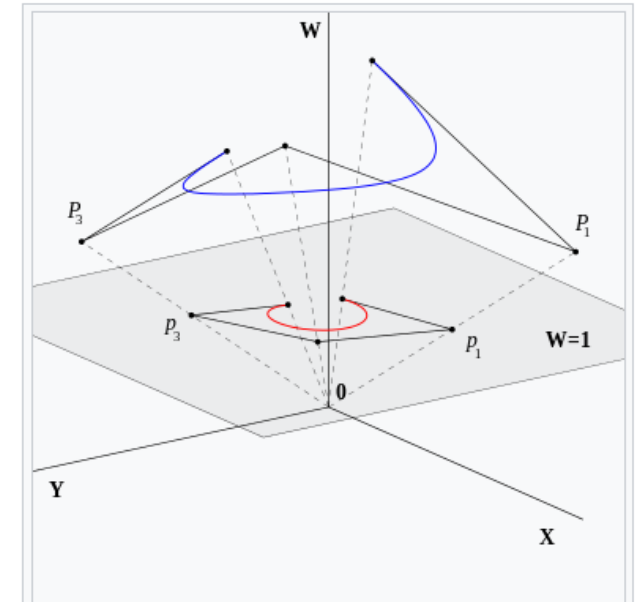
Article [Talk](#)

[Read](#) [Edit](#) [View history](#) [Tools](#) ▼

From Wikipedia, the free encyclopedia

In [mathematics](#), **homogeneous coordinates** or **projective coordinates**, introduced by [August Ferdinand Möbius](#) in his 1827 work *Der barycentrische Calcul*,^{[1][2][3]} are a [system of coordinates](#) used in [projective geometry](#), just as [Cartesian coordinates](#) are used in [Euclidean geometry](#). They have the advantage that the coordinates of points, including [points at infinity](#), can be represented using finite coordinates. Formulas involving homogeneous coordinates are often simpler and more symmetric than their Cartesian counterparts. Homogeneous coordinates have a range of applications, including [computer graphics](#) and 3D [computer vision](#), where they allow [affine transformations](#) and, in general, [projective transformations](#) to be easily represented by a [matrix](#). They are also used in fundamental [elliptic curve cryptography](#) algorithms.^[4]

If homogeneous coordinates of a point are multiplied by a non-zero [scalar](#) then the resulting coordinates represent the same point. Since homogeneous coordinates are also given to points at infinity, the number of coordinates required to allow this extension is one more than the dimension of the [projective space](#) being considered. For example, two homogeneous coordinates are required to specify a point on the projective line and three homogeneous coordinates are required to specify a point in the projective plane.



Rational Bézier curve – polynomial curve defined in homogeneous coordinates (blue) and its projection on plane – rational curve (red)

To go from a familiar 2D pinhole camera projection $[u, v]$ coordinates to 3d homogeneous coordinates, stack them with the Z of the triple reciprocal with the distance from the pinhole to the projection plane along the principal ray for each camera.

To summarize:


- Any point in the projective plane is represented by a triple (X, Y, Z) , called **homogeneous coordinates** or **projective coordinates** of the point, where X , Y and Z are not all 0.
- The point represented by a given set of homogeneous coordinates is unchanged if the coordinates are multiplied by a common factor.
- Conversely, two sets of homogeneous coordinates represent the same point if and only if one is obtained from the other by multiplying all the coordinates by the same non-zero constant.
- When Z is not 0 the point represented is the point $(X/Z, Y/Z)$ in the Euclidean plane.
- When Z is 0 the point represented is a point at infinity.

The triple $(0, 0, 0)$ is omitted and does not represent any point.

The [origin](#) of the Euclidean plane is represented by $(0, 0, 1)$.

Each point in the Pinhole Camera 2D projection space $[u,v]$ has a magnification factor k given the depth of field of the source voxel M_i ,

The signal measured at m_i is a line integral through a set of voxels in the 3D homogeneous coordinate projection space volume, which is equivalent to the source volume with a known image-to-image transform for each camera



PyTorch
geometry

v0.1.2

Search docs

PACKAGE REFERENCE

torchgeometry.core

Image Transformations

Linear Transformations

Pinhole Camera

Conversions

Warping

torchgeometry.image

torchgeometry.losses

torchgeometry.contrib

torchgeometry.utils

TUTORIALS

Rotate image using warp affine transform

Warp image using perspective transform

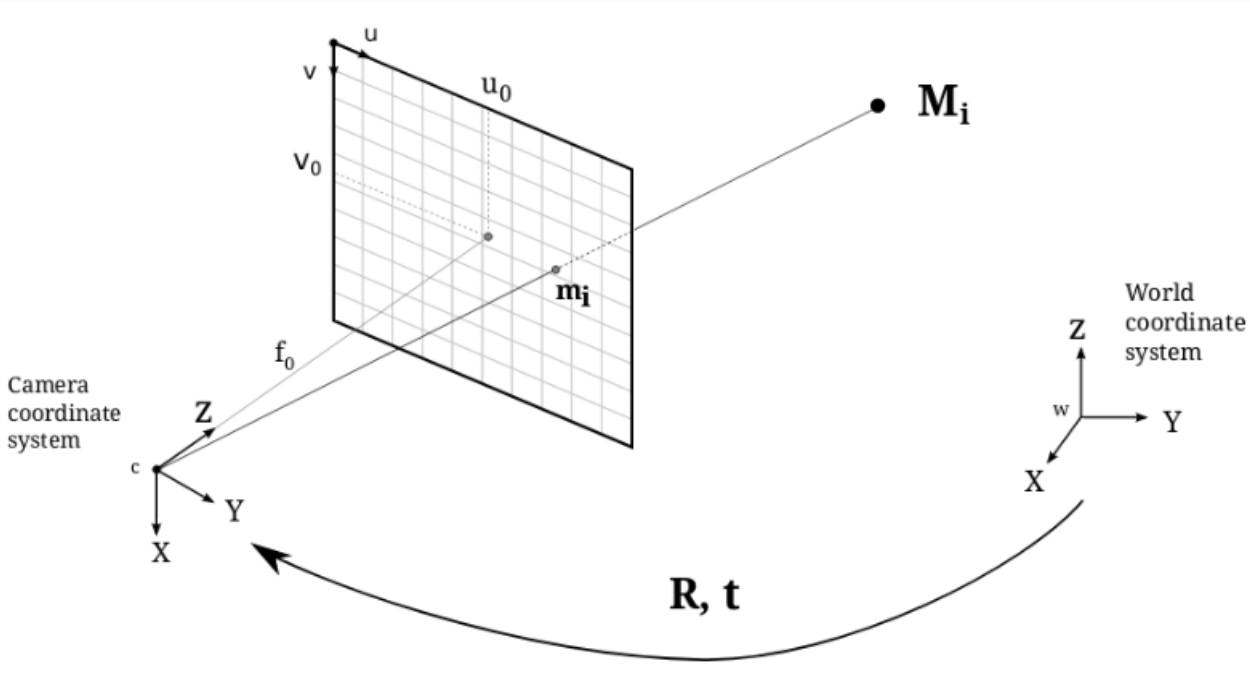
Blur image using GaussianBlur operator

Pinhole Camera

In this module we have all the functions and data structures needed to describe the projection of a 3D scene space onto a 2D image plane.

In computer vision, we can map between the 3D world and a 2D image using *projective geometry*. The module implements the simplest camera model, the **Pinhole Camera**, which is the most basic model for general projective cameras from the finite cameras group.

The Pinhole Camera model is shown in the following figure:

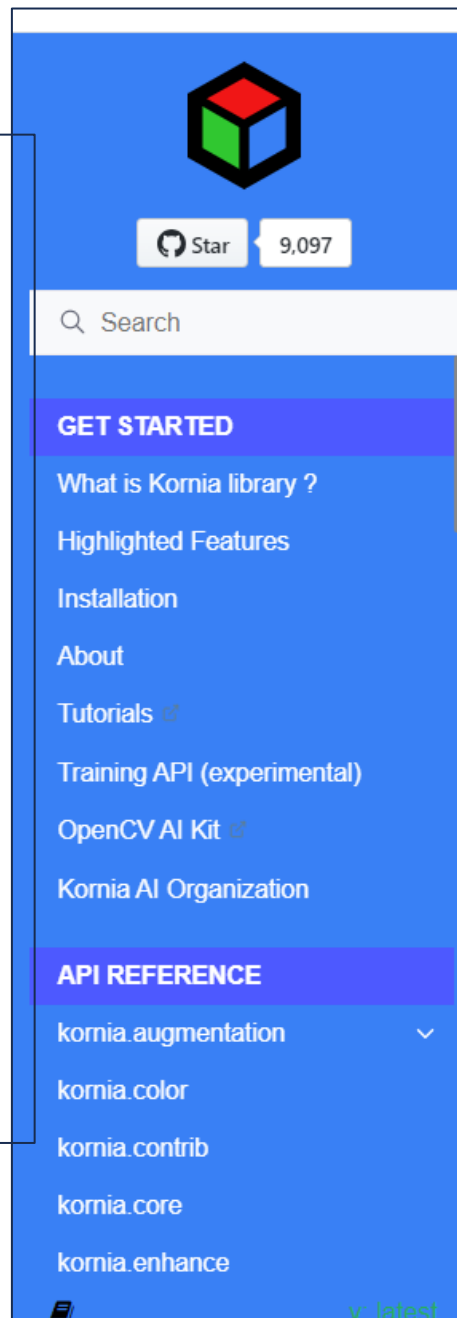


Using this model, a scene view can be formed by projecting 3D points into the image plane using a

There is lots of open source software for computer vision, including for 3D vision.

Most of it, however, concerns depth maps of occulting volumes with surfaces, because it is using visible light cameras.

For us, the volume is transparent enough where we can reconstruct throughout the entire 3D volume.



Kornia



Open Source and Computer Vision

Powered by  PyTorch

State-of-the-art and curated Computer Vision algorithms for AI.

Kornia AI is on the mission to leverage and democratize the next generation of Computer Vision tools and Deep Learning libraries within the context of an Open Source community.

```
>>> import kornia.geometry as K
>>> registrator = K.ImageRegistrator('similarity')
>>> model = registrator(img1, img2)
```

Ready to use with state-of-the art Deep Learning models:


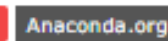
```
>>> import torch.nn as nn
>>> import kornia.contrib as K
>>> classifier = nn.Sequential(
...     K.VisionTransformer(image_size=224, patch_size=16),
...     K.ClassificationHead(num_classes=1000),
... )
>>> logits = classifier(img)    # BxN
>>> scores = logits.argmax(-1) # B
```



PyTorch3D is related to pytomography and diffDRR open source packages

With our known geometry it is simpler to just perform the linear algebra directly rather than to call and use their methods, at least for now. We do not require a mesh render, for example – our situation is different.



 FAILED  0.7.5

Introduction

PyTorch3D provides efficient, reusable components for 3D Computer Vision research with [PyTorch](#).

Key features include:

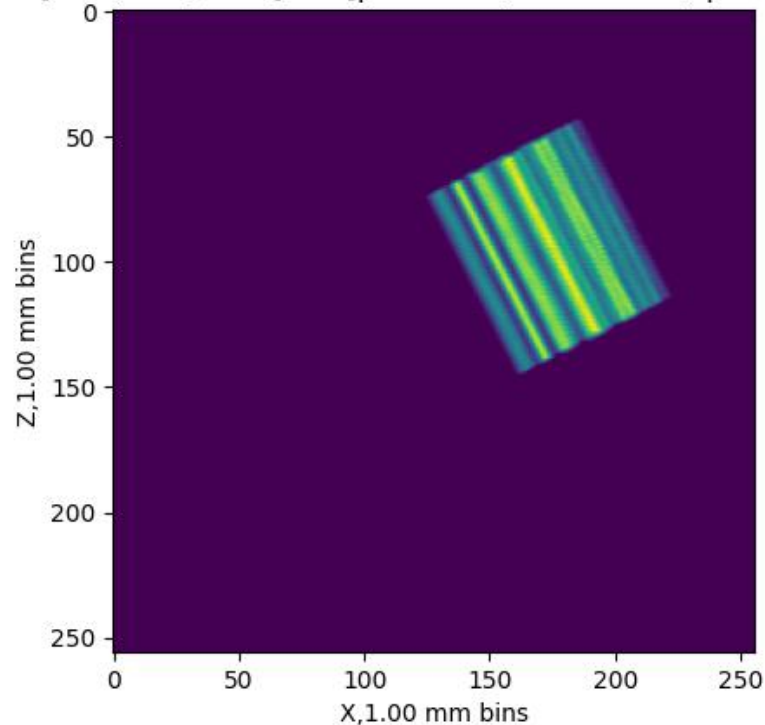
- Data structure for storing and manipulating triangle meshes
- Efficient operations on triangle meshes (projective transformations, graph convolution, sampling, loss functions)
- A differentiable mesh renderer
- Implicitron, see [its README](#), a framework for new-view synthesis via implicit representations. ([blog post](#))

PyTorch3D is designed to integrate smoothly with deep learning methods for predicting and manipulating 3D data. For this reason, all operators in PyTorch3D:

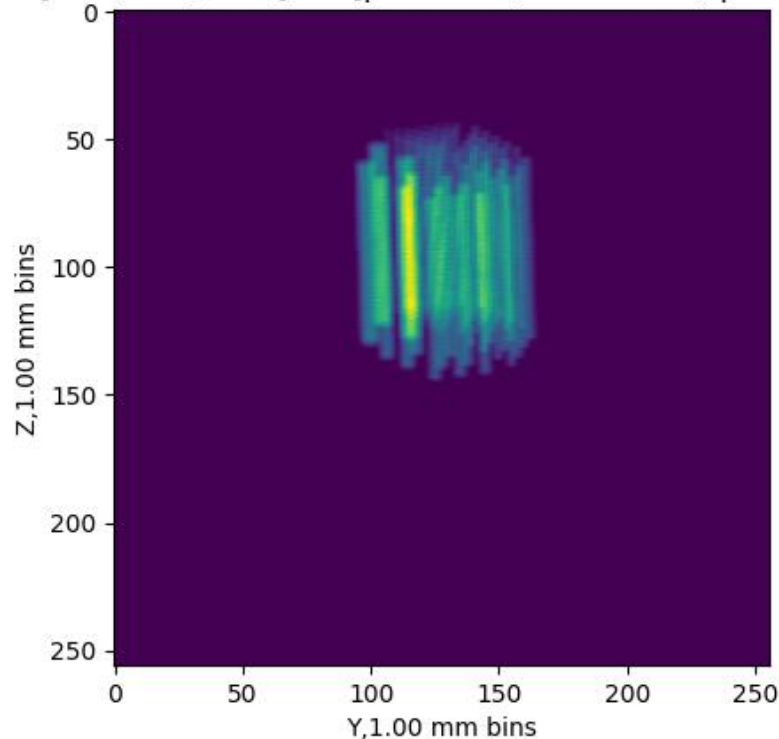
- Are implemented using PyTorch tensors
- Can handle minibatches of heterogeneous data
- Can be differentiated
- Can utilize GPUs for acceleration

Orthogonal summed projection (Euclidean volume) for first random event, with translations and rotations indicated

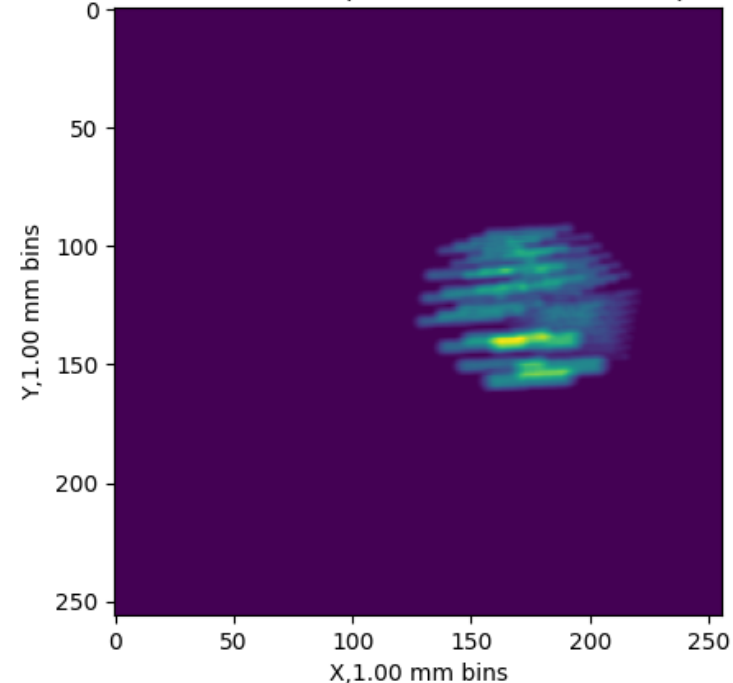
T= [45.9, -0.0, 33.5] R= [psi=285.0, theta=27.2, phi=183.3]



T= [45.9, -0.0, 33.5] R= [psi=285.0, theta=27.2, phi=183.3]



T= [45.9, -0.0, 33.5] R= [psi=285.0, theta=27.2, phi=183.3]



2D summed projections from
homogeneous projection space volume
for first module first random event, with
translations and rotations indicated, still
being debugged

