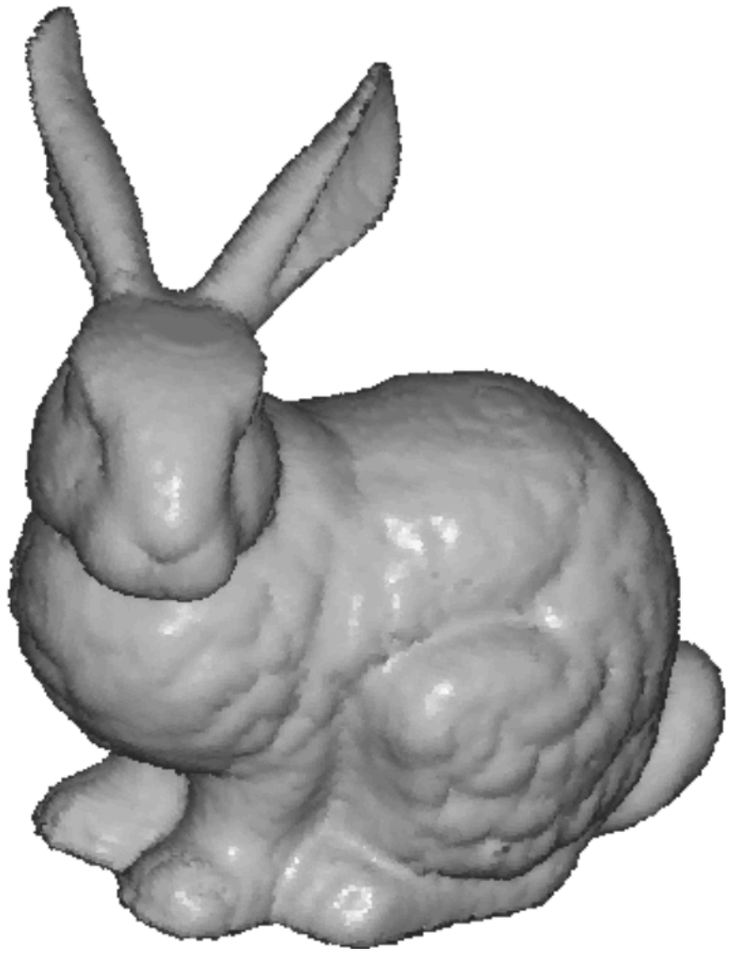
3D* meets neural networks

Practical dive

* no 3DConv today - they are slow and boring

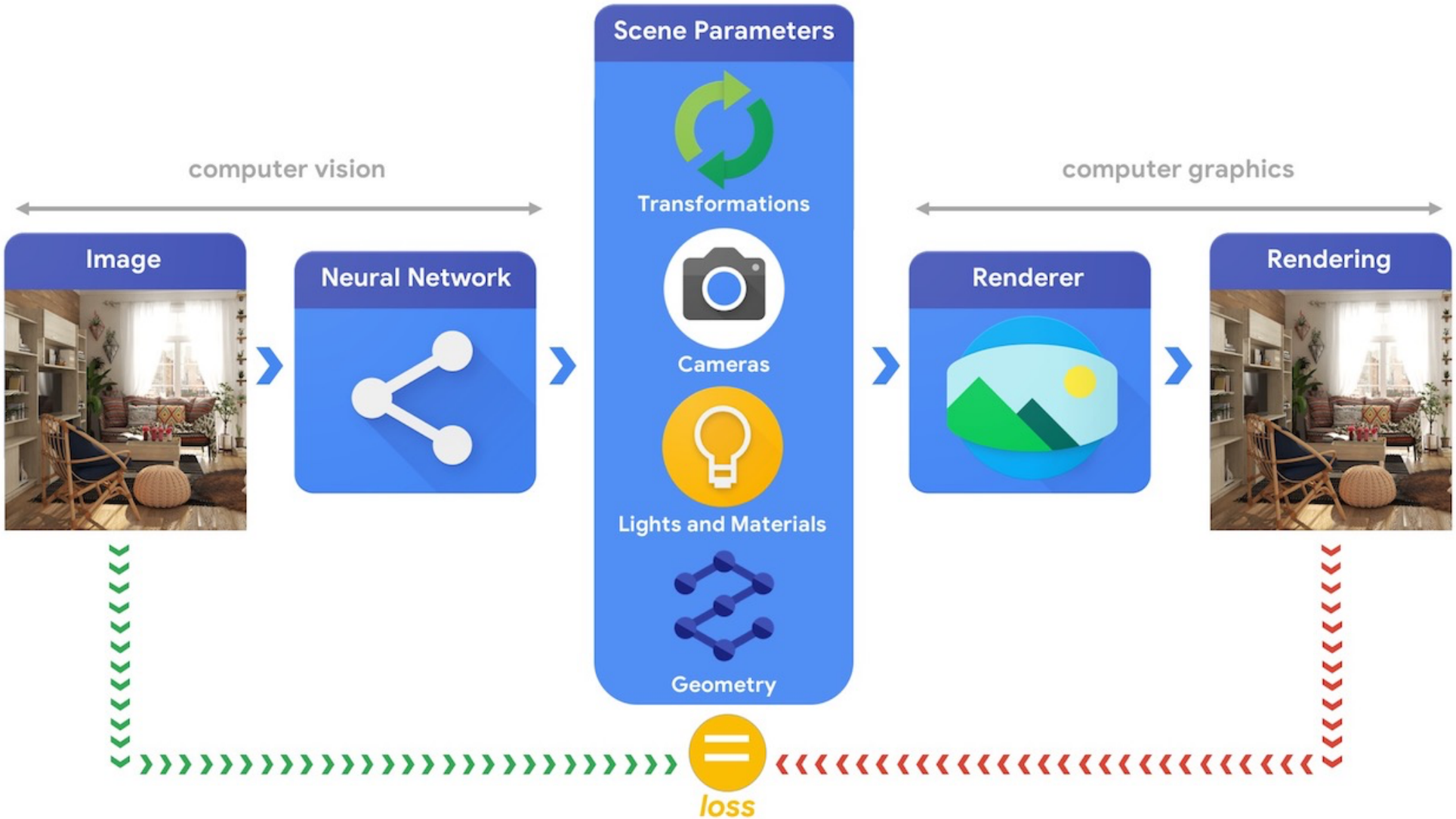
How I got the idea.

+



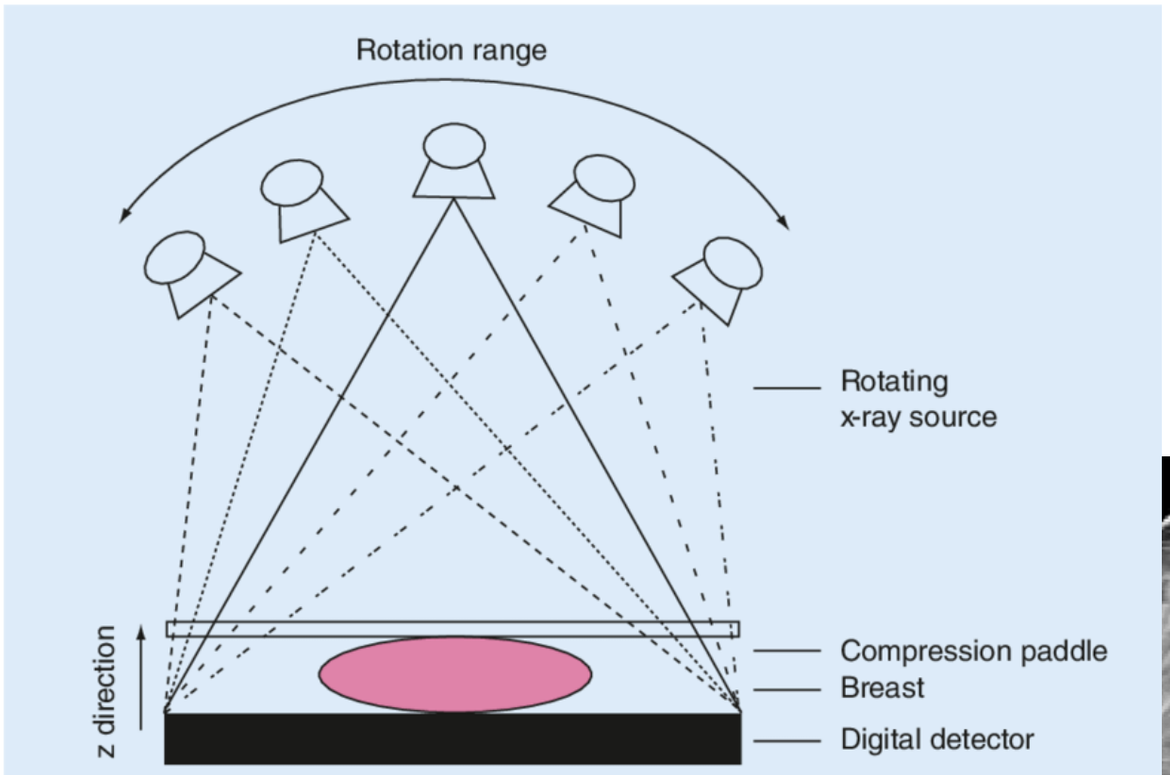
https://github.com/tensorflow/graphics

How I got the idea.



https://github.com/tensorflow/graphics

Tomosynthesis



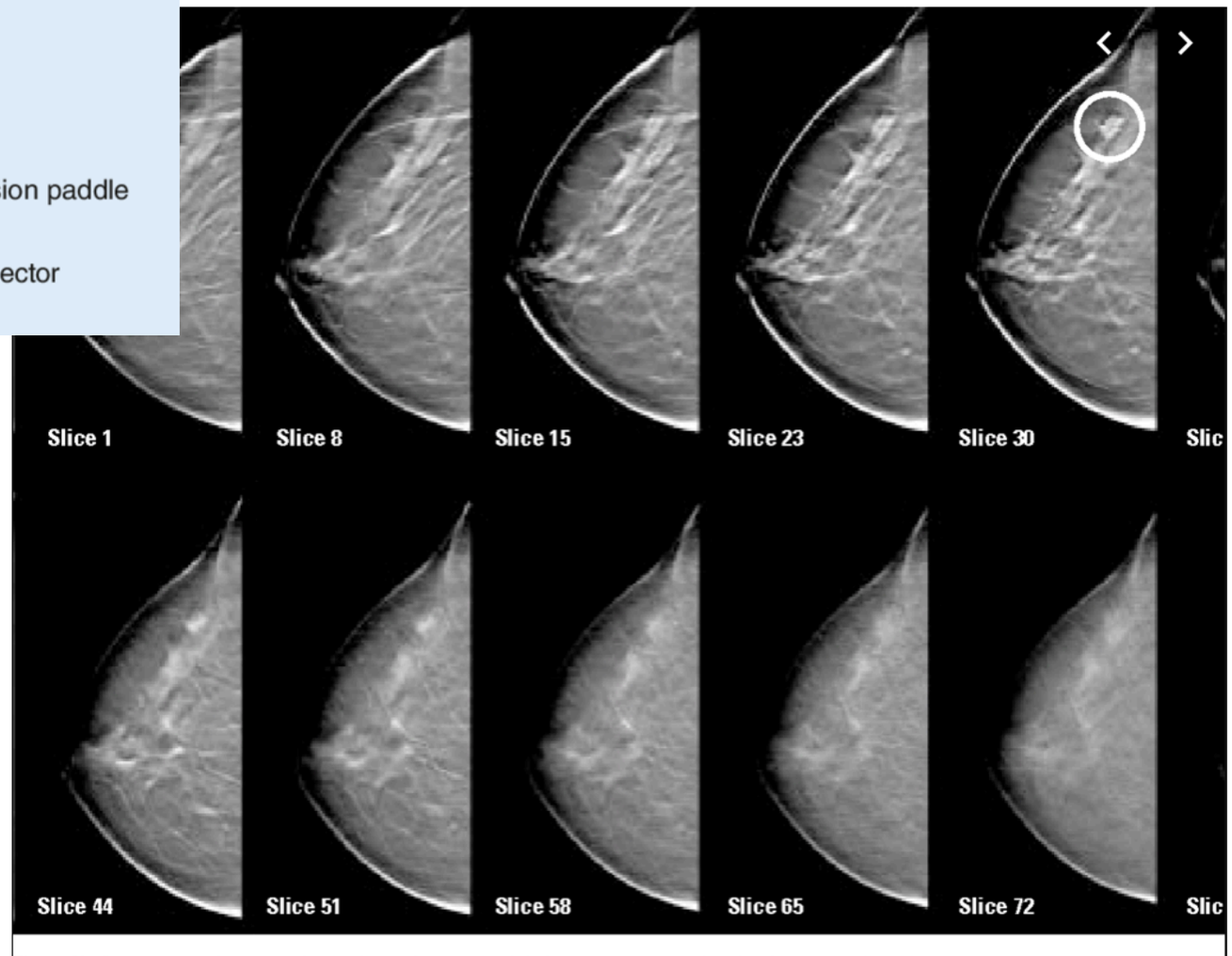


Figure 6b: Reconstructed tomosynthesis slices. An invasive lobular carcinoma can be clearly seen in slice 30

Problem:

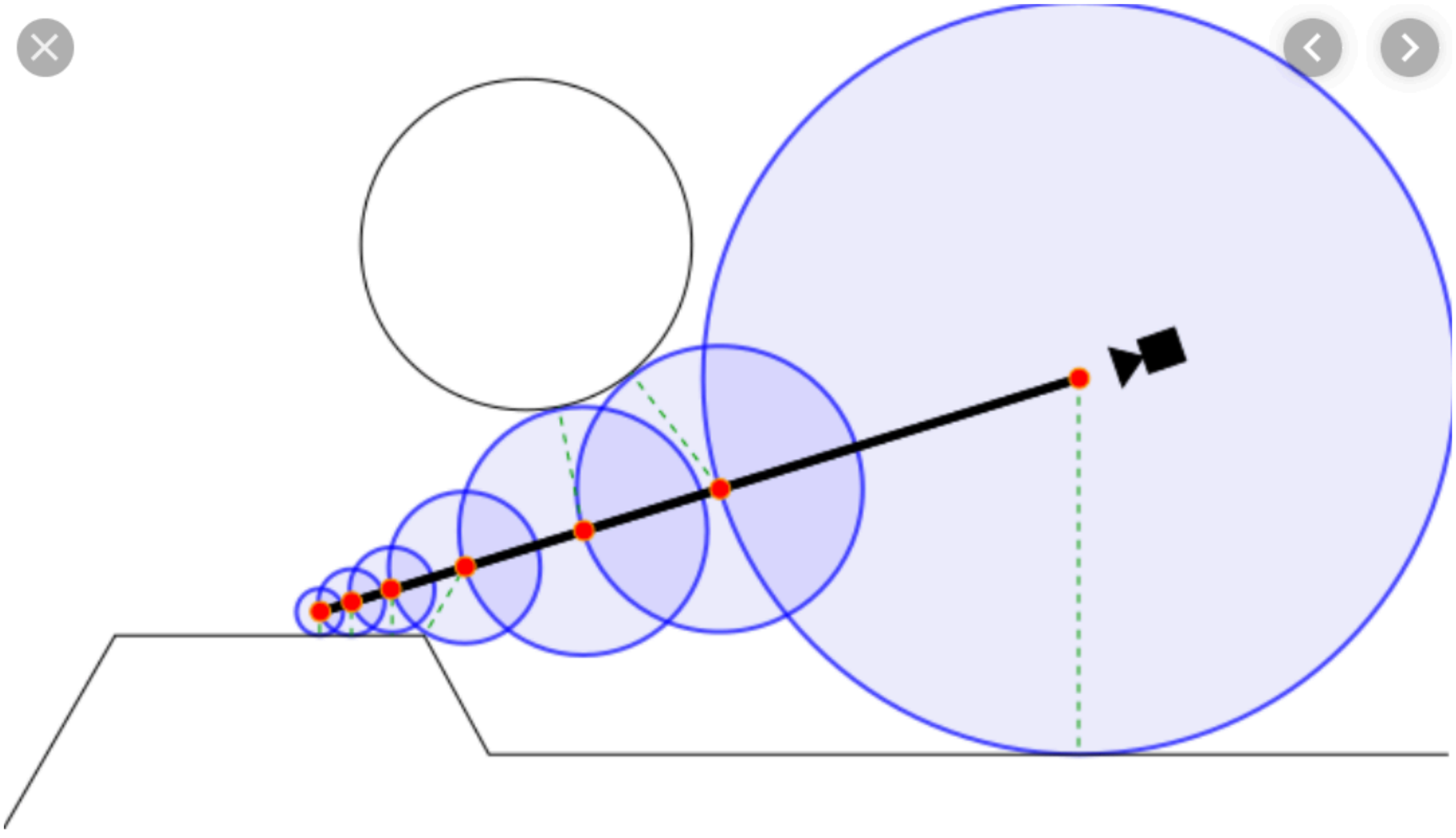
How to reconstruct solid 3D object having no ground truth from a series of 2D projections?

Funny GIF

Key take-ways

- PyTorch is not just for convnets and other hype
- pdb.pm() or %debug
- torch.einsum()
- Nobody’s perfect - torch.norm()
- Ipywidgets
- %timeit

Raymarching



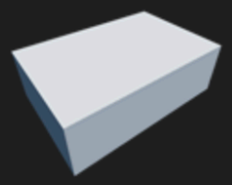
Signed Distance Functions

Function that returns distance to object from point. Signed.

Sphere - exact

```
float sdSphere( vec3 p, float s )
{
  return length(p)-s;
}
```

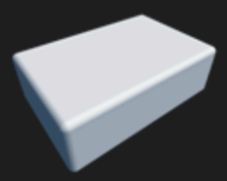
Box - exact



Youtube Tutorial on formula derivation: https://www.youtube.com/watch?v=62-pRVZuS5c

```
float sdBox( vec3 p, vec3 b )
{
  vec3 q = abs(p) - b;
  return length(max(q,0.0)) + min(max(q.x,max(q.y,q.z)),0.0);
}
```

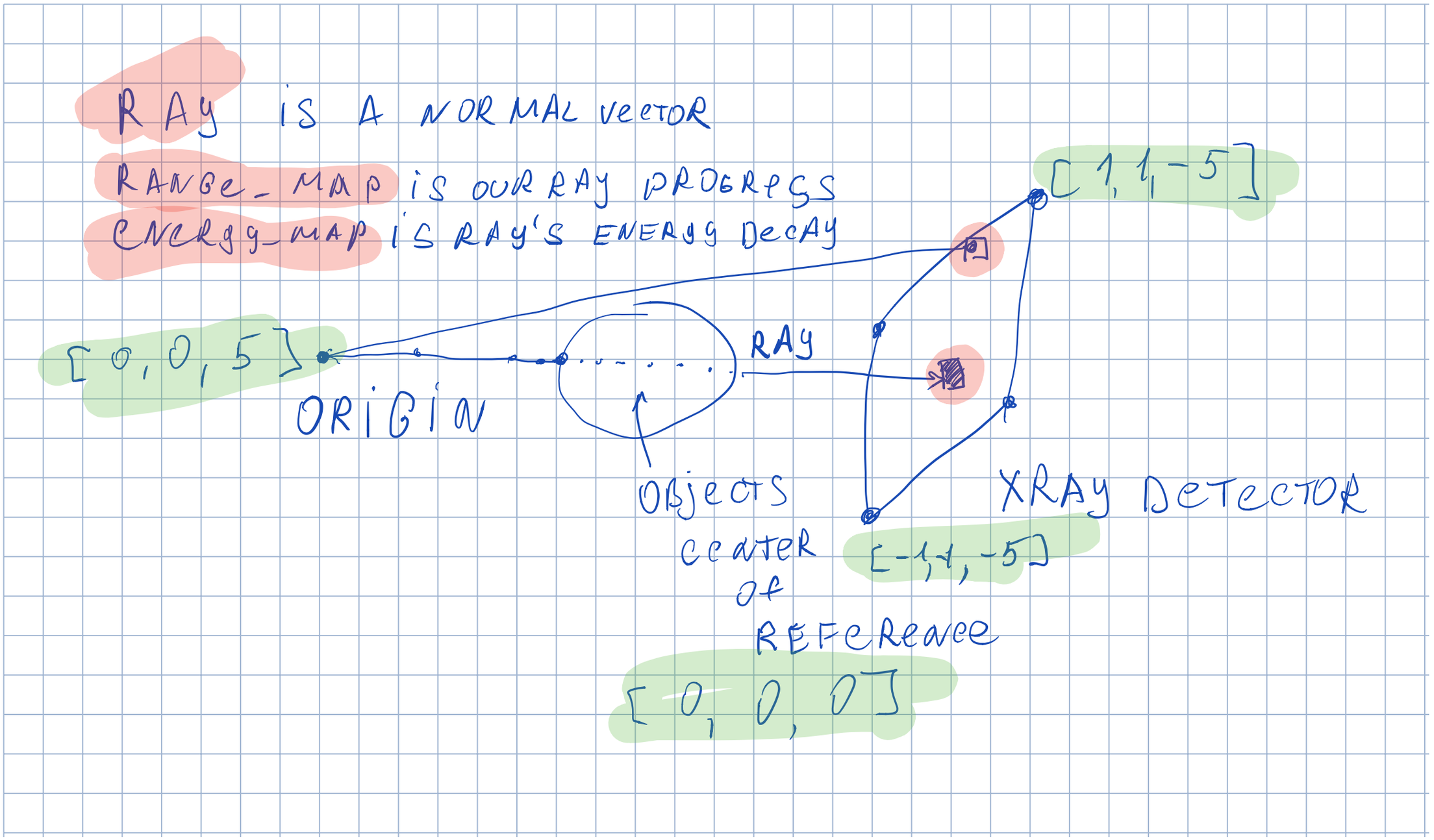
Round Box - exact



```
float sdRoundBox( vec3 p, vec3 b, float r )
{
  vec3 q = abs(p) - b;
  return length(max(q,0.0)) + min(max(q.x,max(q.y,q.z)),0.0) - r;
}
```

https://www.iquilezles.org/www/articles/distfunctions/distfunctions.htm

Today's algorithm invariant



Ok, let’s do it

Open notebook 0.

- **create Detector Square - thing that acts as a sensor**
- **create origin - point that emits rays**
- **create Rays - vectors connecting origin to sensor**
- **marsh them using objects SDF**

Cool. Now we need volume.
Notebook 1

Few options to get “volume”.

- **make “light”**
- **make “opacity”**

We stick to opacity, due to my personal interest in X-rays.

Now we convert this stuff to torch and remove loops. Notebook 2.

Direction vector from two points

Given that a vector $\overrightarrow{PQ}$ has an initial point at $P(2, 2, 1)$ and a terminal point at $Q(6, 3, 2)$, find the vector $\overrightarrow{PQ}$:

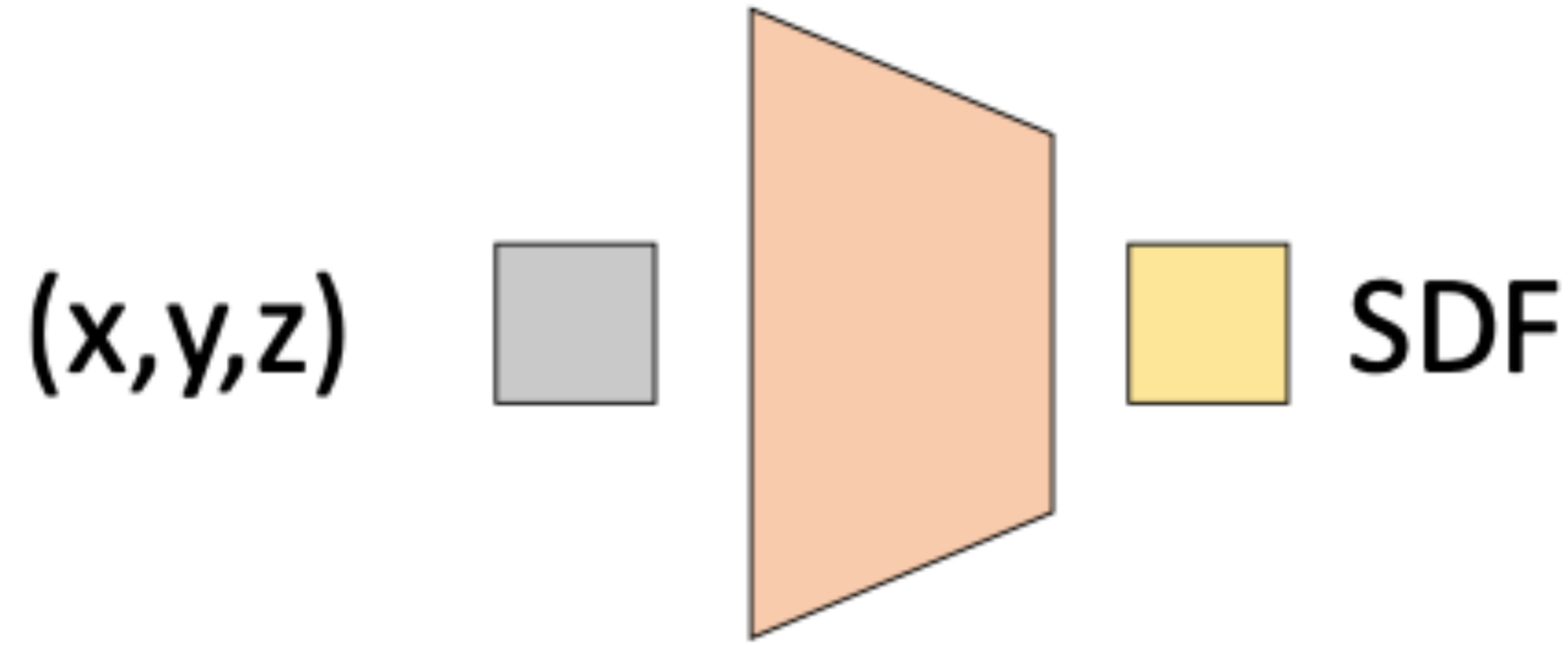
To do this, we will simply subtract point P from point Q to obtain:

$$\overrightarrow{PQ} = (x_Q - x_P, y_Q - y_P, z_Q - z_P)$$
$$\overrightarrow{PQ} = (6 - 2, 3 - 2, 2 - 1)$$
$$\overrightarrow{PQ} = (4, 1, 1)$$

Notebooks to 5.

Complex scenes
Ipywidgets
Rotation matrix
Einsum

Notebook 6-7. SDF approximation



Notebook 8.

We got a result. Try your own scene.

Notebook 9+ versions.