Planar and Spherical Projections of a Point Cloud (Using Open3D)

A point cloud is a collection of points in 3-dimensional coordinate space represented by x, y, and z axes. These points may just form a geometric shape or more complex structures such as a 3-dimensional representation of the real world. Various libraries, even with open-sources, such as the Point Cloud Library (PCL) and Intel's Open3D, provide native datatype for a point cloud. These libraries not only allow simple geometric transformations of the cloud but also provide sophisticated functionalities such as the reconstruction of a 3D scene from various images.

However, this article discusses a simple geometric approach to achieve the reverse of 3D reconstruction, i.e generate an image from a point cloud. We will also see how to project the point cloud onto other non-planar geometries such as a sphere. Hence, our objective is the following

- Project a point cloud from a certain perspective to a given plane, then store the projection as an image; and
- Project the point cloud onto the surface of a given sphere.

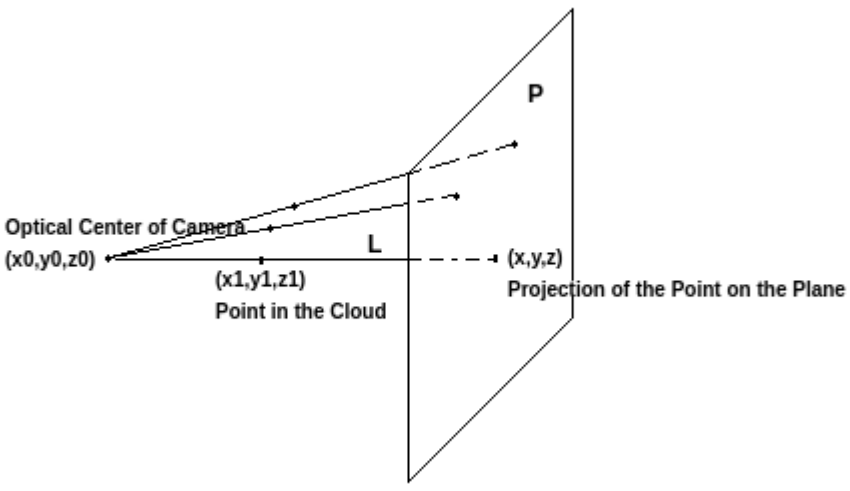
The sample implementation included in this article was run in python3.6. The following additional libraries for python must be available

- Open3D (tested with v0.9)
- NumPy

Projection onto a Plane

We will try to achieve the perspective projection of a point cloud (i.e. from a fixed optical center) onto a plane given by the general equation:

$$Ax + By + Cz + D = 0$$



The above demonstration shows the perspective transform of a set of points onto the plane P from the optical center of the camera, with arbitrary co-ordinates (x_0, y_0, z_0). We can see that the projection (x, y, z)of a point (x_1, y_1, z_1) on the plane P is nothing but the point of intersection between the said plane and a line L passing through the optical center and the given point in the cloud.

The line L in 3-dimensional space can be represented as follows

$$x = x_1 + at$$
$$y = y_1 + bt$$
$$z = z_1 + ct$$

where a, b and c are the direction cosines of the line L, and t is known as the direction ratio.

With the given optical center and the set of points in the cloud, we can compute the direction cosines for the lines passing through each of the points.

$$a = cos\alpha = \frac{x_1 - x_0}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}}$$
$$b = cos\beta = \frac{y_1 - y_0}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}}$$
$$c = cos\gamma = \frac{z_1 - z_0}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}}$$

We can substitute the equations of the line in the equation of the plane P since we are solving for the point of intersection between the two.

$$A(x_1 + at) + B(y_1 + bt) + C(z_1 + ct) + D = 0$$

Solving for t, we can deduce the above equation as follows

$$t = -\frac{Ax_1 + By_1 + Cz_1 + D}{aA + bB + cC}$$

Finally, we can obtain the projection by substituting the value of t into the equations of the line L.

$$x = x_1 - a * \frac{Ax_1 + By_1 + Cz_1 + D}{aA + bB + cC}$$
$$y = y_1 - b * \frac{Ax_1 + By_1 + Cz_1 + D}{aA + bB + cC}$$
$$z = z_1 - c * \frac{Ax_1 + By_1 + Cz_1 + D}{aA + bB + cC}$$

The above computation should be performed for each point in the Point Cloud to obtain the complete projection on the given plane.

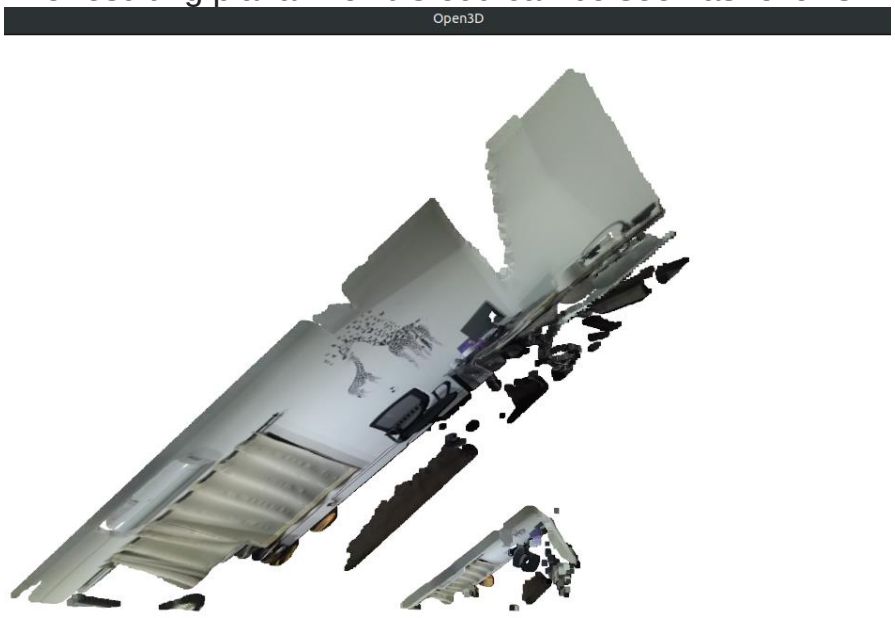
Python Implementation

The following function takes an Open3D PointCloud, equation of a plane (A, B, C, and D) and the optical center and returns a planar Open3D PointCloud Geometry.

```
def get_flattened_pcds2(source,A,B,C,D,x0,y0,z0):
    x1 = np.asarray(source.points)[:,0]
    y1 = np.asarray(source.points)[:,1]
    z1 = np.asarray(source.points)[:,2]
    x0 = x0 * np.ones(x1.size)
    y0 = y0 * np.ones(y1.size)
    z0 = z0 * np.ones(z1.size)
    r = np.power(np.square(x1-x0)+np.square(y1-y0)+np.square(z1-z0),0.5)
    a = (x1-x0)/r
    b = (y1-y0)/r
    c = (z1-z0)/r
    t = -1 * (A * np.asarray(source.points)[:,0] + B * np.asarray(source.points)[:,1] + C *
np.asarray(source.points)[:,2] + D)
    t = t / (a*A+b*B+c*C)
    np.asarray(source.points)[:,0] = x1 + a * t
    np.asarray(source.points)[:,1] = y1 + b * t
    np.asarray(source.points)[:,2] = z1 + c * t
    return source
```

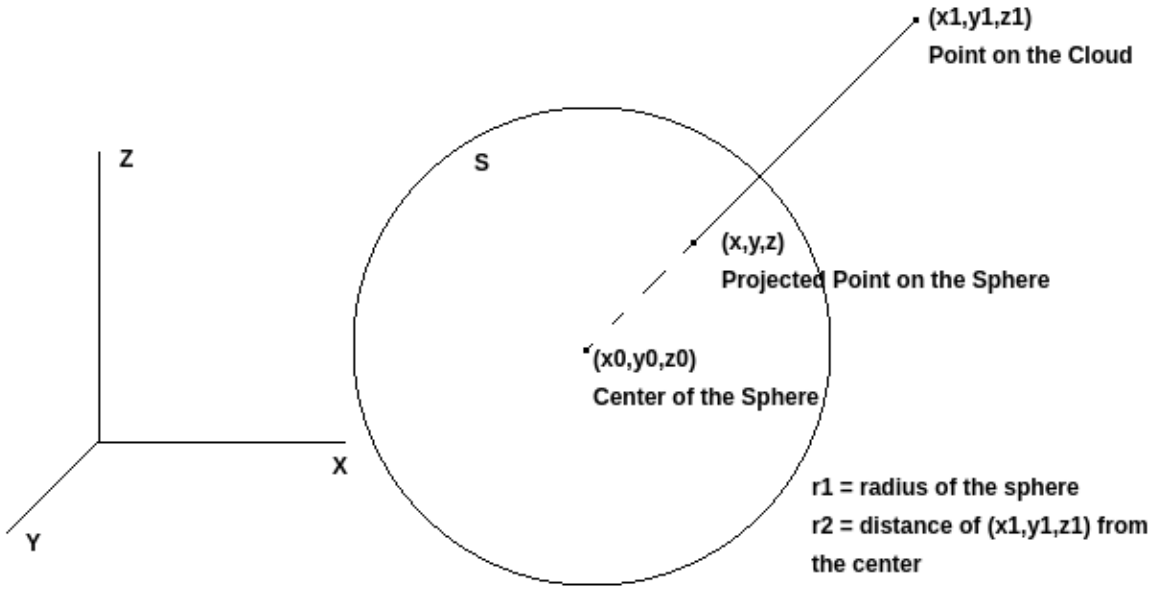
Output

The resulting planar Point Cloud can be seen as follows



Projection onto a Sphere

A Point Cloud maybe projected onto geometrical surfaces other than a plane, such as a sphere. The spherical projection in particular maybe used to approximate a 360 image of the Point Cloud since those are spherical or cylindrical photographs of a scene.



The figure above shows the spherical projection of a point onto the surface of a sphere with radius r_1 and with its center at the point(x_0, y_0 , z_0). Such a sphere can be represented by the following equation

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r_1^2$$

Similar to the projection on a plane, the spherical projection of a point is the point of intersection between the given sphere S and the line passing through the center of the sphere and the point.

Again, the line L in 3-dimensional space can be represented as follows

$$x = x_1 + at$$
$$y = y_1 + bt$$
$$z = z_1 + ct$$

where a, b and c are the direction cosines of the line, and t is the direction ratio. With the given center of the sphere and the set of points in the cloud, we can compute the direction cosines for the lines passing through each of the points and the center.

Here,

$$r_2 = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}$$

is the distance between the center of the sphere and the point in the Point Cloud.

Now, we can substitute the equations of the line into the equations of the sphere to solve for t.

$$(x_1 + at - x_0)^2 + (y_1 + bt - y_0)^2 + (z_1 + ct - z_0)^2 = r_1^2$$

Solving this equation for t, we obtain the following relation.

$$t = r_1 - r_2$$

Finally, substituting this value of t and the direction cosines into the equations of the line, we can obtain the projected points.

$$x = x_1 + at = x_1 + \frac{x_1 - x_0}{r_2}(r_1 - r_2)$$

or,

$$x = x_0 + \frac{r_1}{r_2}(x_1 - x_0)$$
$$y = y_0 + \frac{r_1}{r_2}(y_1 - y_0)$$
$$z = z_0 + \frac{r_1}{r_2}(z_1 - z_0)$$

Python Implementation

The following python method takes an Open3D PointCloud geometry, and the radius and center of the sphere to project on.

```
def get_spherical_pcd(source,x0,y0,z0,r1):
    x2 = np.asarray(source.points)[:,0]
    y2 = np.asarray(source.points)[:,1]
    z2 = np.asarray(source.points)[:,2]
    x0 = x0 * np.ones(x2.size)
    y0 = y0 * np.ones(y2.size)
    z0 = z0 * np.ones(z2.size)
    r1 = r1 * np.ones(x2.size)
    r2 = np.power(np.square(x2-x0)+np.square(y2-y0)+np.square(z2-z0),0.5)
    np.asarray(source.points)[:,0] = x0 + (r1/r2) * (x2-x0)
    np.asarray(source.points)[:,1] = y0 + (r1/r2) * (y2-y0)
    np.asarray(source.points)[:,2] = z0 + (r1/r2) * (z2-z0)
    return source
```

Output

The resulting spherical projection of the cloud is as follows

References

https://blog.ekbana.com/planar-and-spherical-projections-of-a-point-cloud-d796db76563e