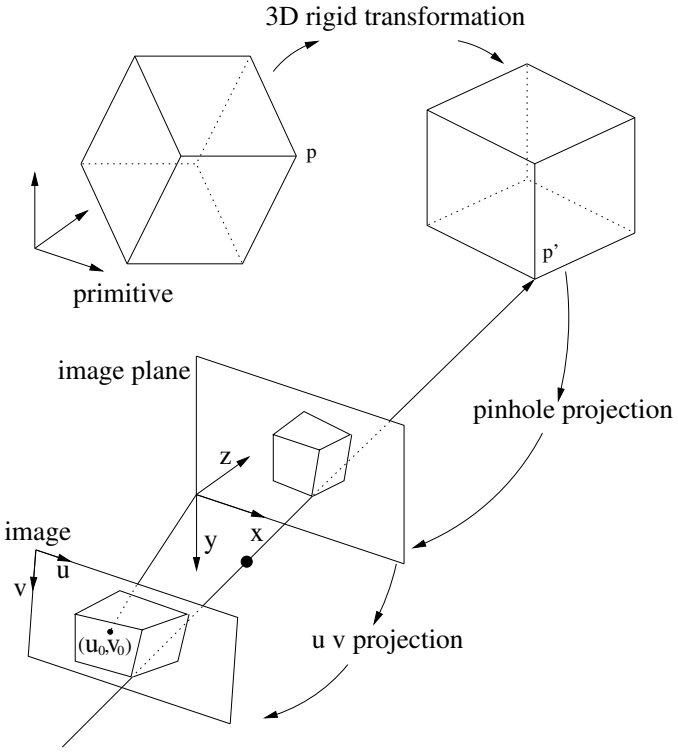
TP5
Image Formation

The objective of this practical work is to study image formation by projecting a 3D point cloud to an image plane using the pinhole camera model.

1 The method

1.1 Pipeline overview



As shown in Figure 1.1, to form an image, one needs first to place an object (the cube) properly in front of the camera by applying rigid transformation to it. Then the object is projected through

pinhole to the image plane. And finial the image plane is fitted in uv coordinates (pixel coordinates) in an image. The 3D rigid transformation, pinhole projection and uv projection will be explained below.

1.2 Rigid transform in 3D-space

Transform of a point in 3D-space:

$$\begin{pmatrix} X_{p'} \\ Y_{p'} \\ Z_{p'} \\ 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix} \tag{1}$$

where $\boldsymbol{R}$ is 3×3 rotational matrix, t a 3×1 translation vector, and $\boldsymbol{p'}$ the new point position after transform.

In practice 3D rotation can be represented in many ways (such as Euler angle, quaternion, and axis-angle) and the calculation of the rotation matrix out of those representations varies. In this TP session we would use yaw, pitch, and roll representation. The code to compute the transformation matrix is provided, which takes the three rotation angles and a translation vector as input.

1.3 Pinhole projection

Perspective projection with the origin in the image plane:

$$\begin{pmatrix} X_{p'} \\ Y_{p'} \\ 0 \\ 1 + Z_{p'}/f \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/f & 1 \end{pmatrix} \begin{pmatrix} X_{p'} \\ Y_{p'} \\ Z_{p'} \\ 1 \end{pmatrix} \tag{2}$$

and thus:

$$\begin{pmatrix} X_{img} \\ Y_{img} \end{pmatrix} = \begin{pmatrix} X_{p'}/(1 + Z_{p'}/f) \\ Y_{p'}/(1 + Z_{p'}/f) \end{pmatrix} \tag{3}$$

1.4 Pixel coordinates u v projection

Now we need to fit a rectangular image to the image plane by choosing the image origin $\begin{pmatrix} u_0 \\ v_0 \end{pmatrix}$ and image resolution α_u and α_v with the unit of meter/pixel so that we can get pixel coordinates (u, v) of the projected point:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} X_{img}/\alpha_u + u_0 \\ Y_{img}/\alpha_v + v_0 \end{pmatrix} \tag{4}$$

where $\begin{pmatrix} u_0 \\ v_0 \end{pmatrix}$ is the pixel coordinates of the origin of image plane, α_u and α_v are scaling factor, and α_u/α_v is called aspect ratio, which is 1 for pinhole camera.

1.5 3D data file

There are many file format to store 3D data, such as .ply, .obj, .off, and .stl. What we are going to use is .off. The following is a typical .off file:

```
COFF
6293    9302    0
0.253   3.61    -0.22   243   102   24   255
...
-0.57   2.16    0.24    132   28    55   255
3       0       523     1
...
3       6292    6290    325
```

The first line can be OFF (plain off) or COFF (off with color). The second line specifies the number of vertices, faces and edges in the file. Then comes the point list. In the example above, each point contains xyz coordinates, rgb color and transparency value. Following point list there are face list and edge list, which will not be explained here since they are irrelevant to this TP session. .off file can be visualized using softwares such as meshlab.

A function that reads the .off file and puts the result in a customized C structure (point3d) is already implemented.

2 Implementation

1. A function called *readOff* that reads .off file is already implemented in *imageFormationUtils.c*. Modify your Makefile by adding "imageFormationUtils.o -lm" and include the header file *imageFormationUtils.h* in your code. You can find more information about using the code in *imageFormationUtils.h*.

2. Read in *cubeFrame.off*. Assuming the object is already placed in a nice position ($p = p'$), implement the pinhole projection to project every point in the object to image plane, i.e., implement equation (2) and (3) in 1.3.

3. Implement the uv projection (equation (5) in 1.4) and output the image.

4. Now we want to re-position the cube. Implement rigid transformation to the cube (equation (1) in 1.2) and replace p with p' in the pinhole projection.

5. How does the image change if we modify the following parameters: focal length, object transform, image center, and image resolution.

6. Implement orthogonal projection (you probably need to change the resolution a lot to visualize the result).

7. Now try the 3D points from file *human.off* and implement a method that considers occlusion such that only the front can be seen.