

TP5 Computer Vision

Created by: Louis Choules

Contents

Table of Figure	1
Introduction	1
Implementation	1

Table of Figure

Figure 1 . basic image	2
Figure 2 . resolution 2000x2000 and using orthographic camera with $\alpha = 0.0005$	7
Figure 3 . resolution 500x500 and using orthographic camera with $\alpha = 0.005$	8
Figure 4 . resolution 3000x4500 and in orthographic camera with $\alpha = 0.0005$	9
Figure 5 . resolution 400x600 and in perspective camera with a focal length of 0.5, $\alpha = 0.0005$	10
Figure 6 . resolution 500x500 and in orthographic camera with $\alpha = 0.005$	11
Figure 7 . resolution 500x500 and in orthographic camera with $\alpha = 0.005$, $\beta = 180$	12

Introduction

The objective is to study image formation by projecting the 3D point cloud to an image plane by using a perspective camera and an orthographic camera with different configurations to understand what the effect for each configuration.

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.

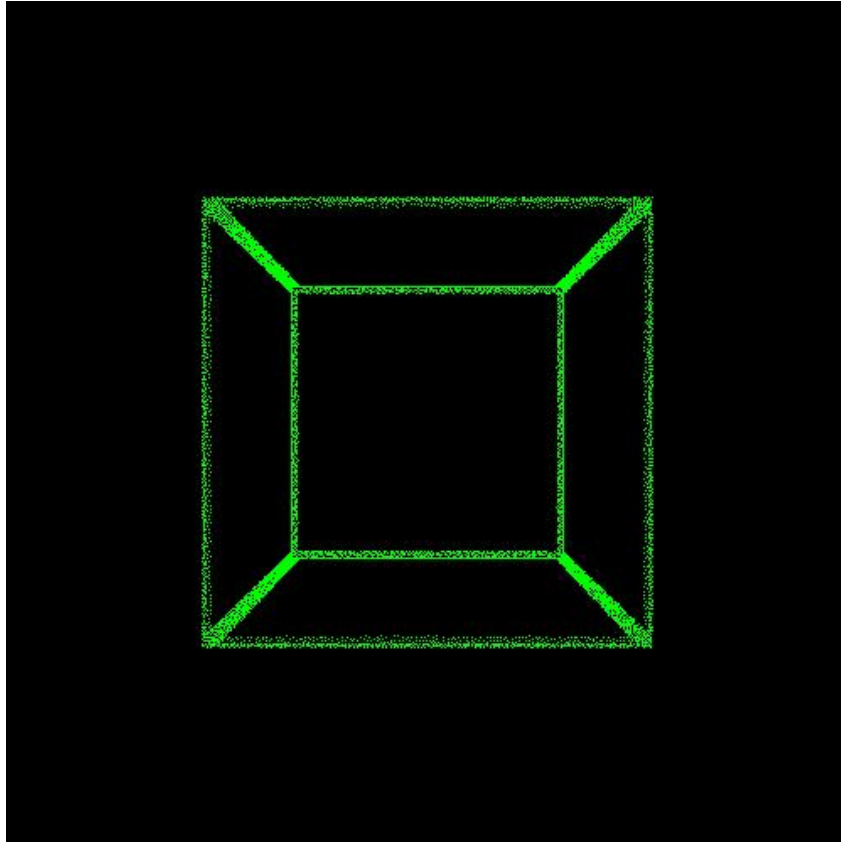
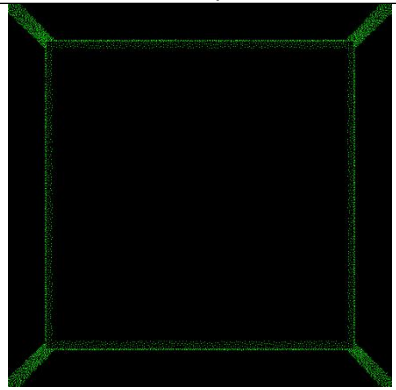


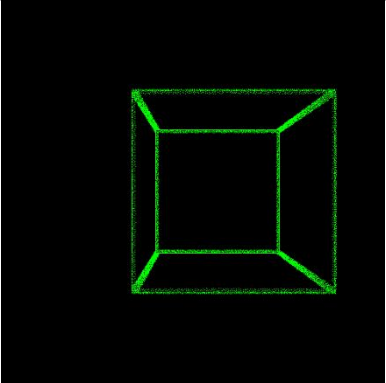
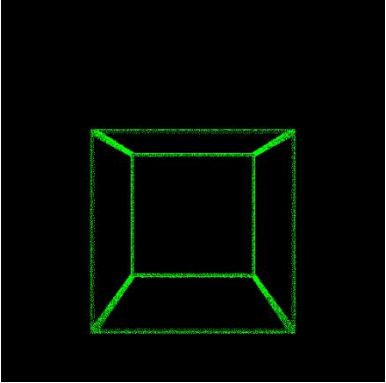
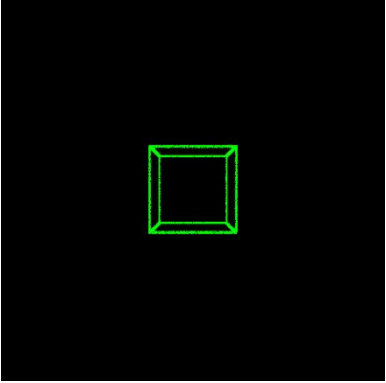
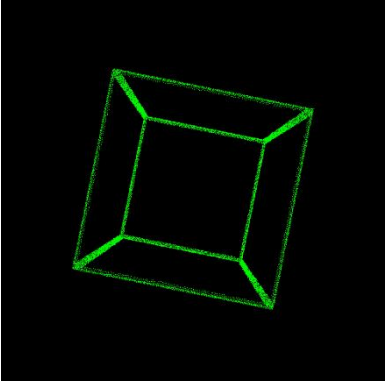
Figure 1. basic image

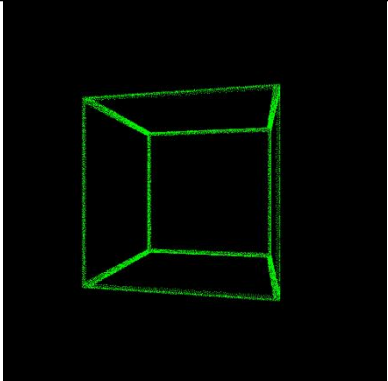
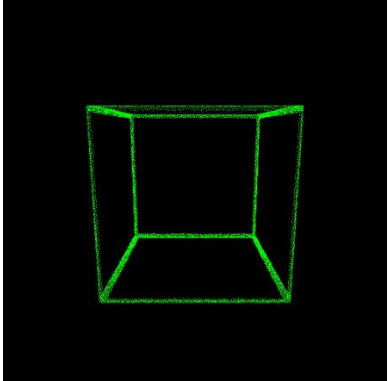
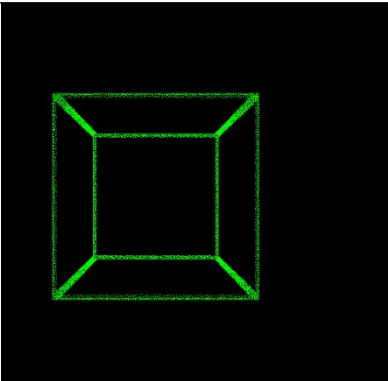
Here is the result of frameCube.off with configuration $f = 0.2$, $\text{resolution_u} = 500$, $\text{resolution_v} = 500$, $\alpha_u = 0.005$, $\alpha_v = 0.005$, $\text{image_origin_u} = 250$ (from $\text{resolution_u}/2$), $\text{image_origin_v} = 250$ (from $\text{resolution_v}/2$).

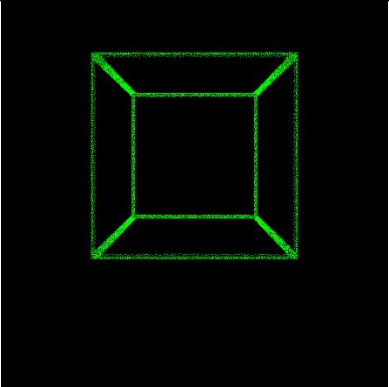
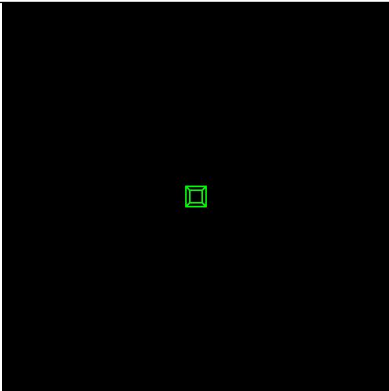
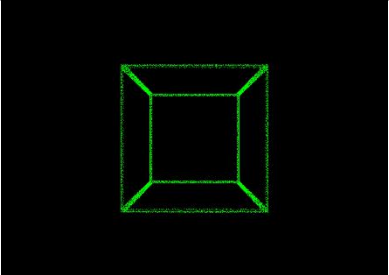
4. Now we want to re-position the cube. Implement rigid transformation to the cube (equation (1) in 1.2) and replace p with p' in tip pinhole projection.

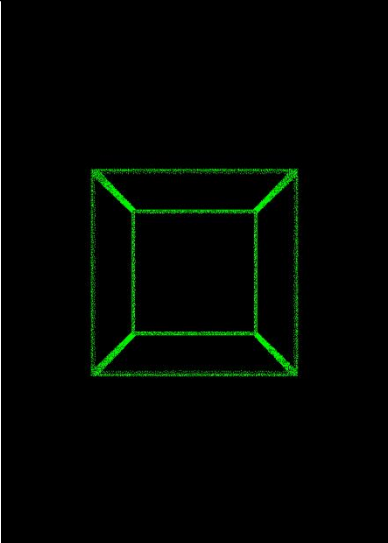
5. How does the image change if we modify the following parameters: focal length, object transform, image center, and image resolution.

Variable	Effect	Example
Focal length	The larger the focal length value will enlarge the object (zoom in). But the smaller the focal length value will decrease the size of the object (zoom out)	 <p>We give 0.5 to the focal length</p>

Transform	X	If the value is larger, the object will move to the right. But if the value is smaller, the object will move to the left	 <p>We give 0.1 to the x-axis</p>
	Y	If the value is larger, the object will move to the below. But if the value is smaller, the object will move to the upper	 <p>We give 0.1 to the y-axis</p>
	Z	If the value is larger, the object will move further away. But if the value is smaller, the object will come closer	 <p>We give 1 to the z-axis</p>
	Alpha	It will rotate the object using the longitudinal axis as the center(roll)	 <p>We give 0.2 to the alpha variable</p>

	Beta	It will rotate the object using the vertical axis as the center (yaw)	 <p>We give 0.2 to the beta variable</p>
	Gama	It will rotate the object using transverse axis as the center (pitch)	 <p>We give 0.2 to the gama variable</p>
Image_center	X	It will move the center of the image on x-axis	 <p>We set the image center to 200 on x-axis (width_image/2– 50)</p>

	γ	It will move the center of the image on the y-axis	 <p>We set the image center to 200 on the y-axis ($\text{height_image}/2 - 50$)</p>
	α	The larger the value, the more point that we will conclude as one point	 <p>We set the α into 0.005</p>
Image_resolution	x	The larger the value, the larger the width of the image	 <p>We give 700 to the image width</p>

	γ	The larger the value, the larger the height of the image	 <p>We give 700 to the image height</p>
--	----------	--	--

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

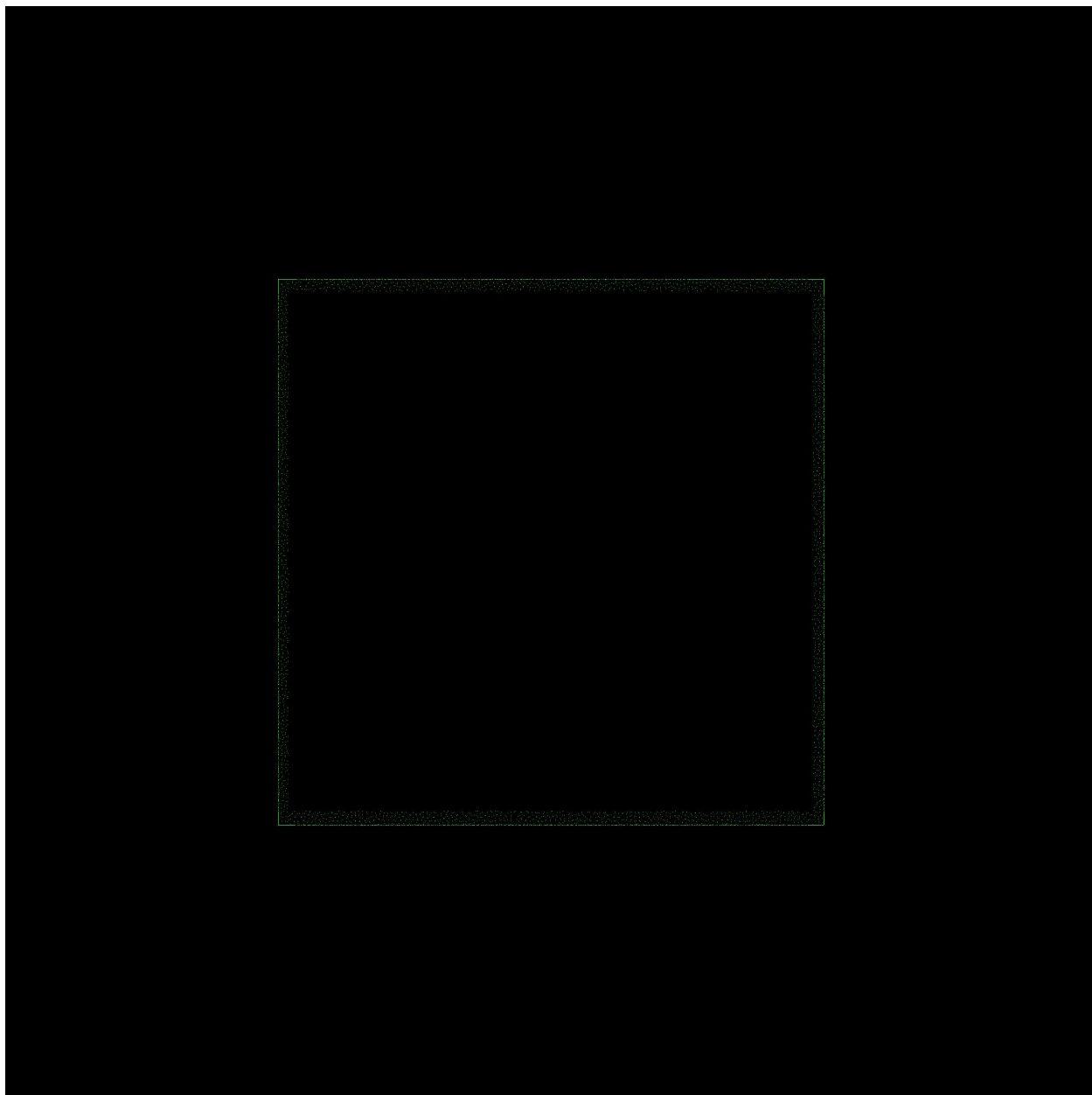


Figure 2. resolution 2000x2000 and using orthographic camera with $\alpha=0.0005$

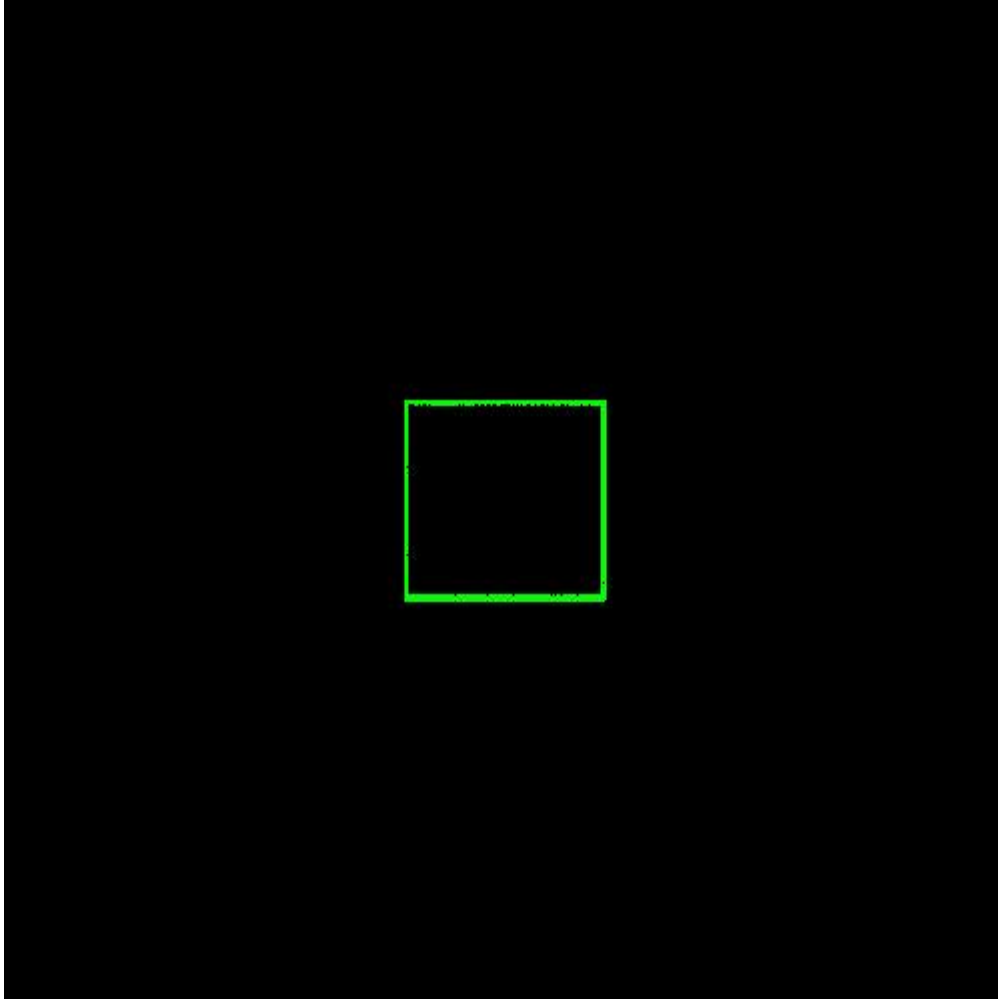


Figure 3. resolution 500x500 and using orthographic camera with $\alpha = 0.005$

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

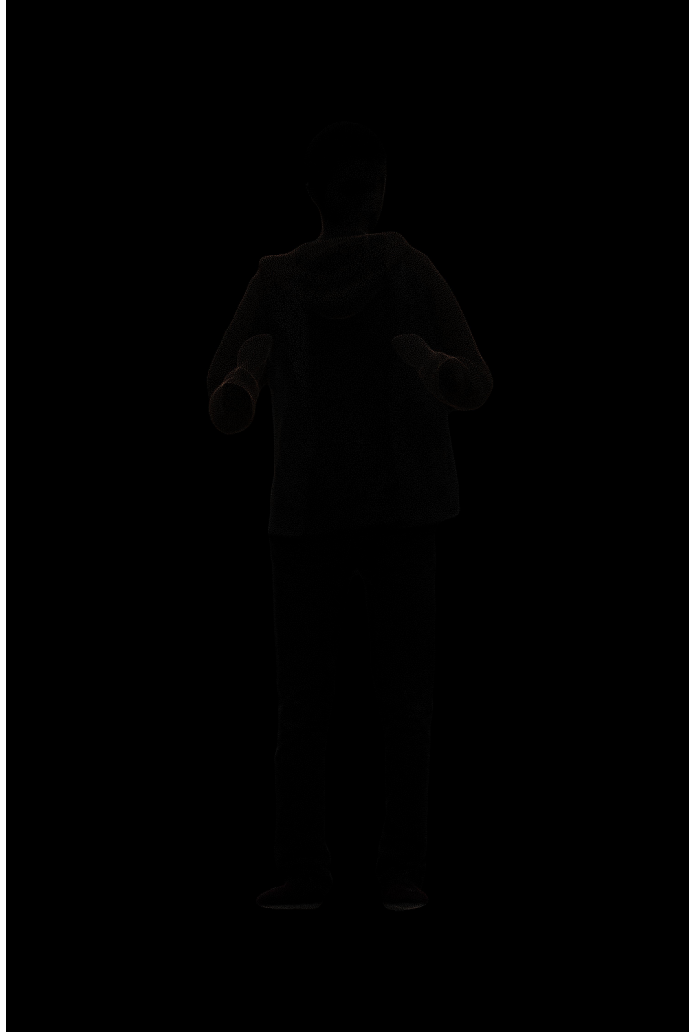


Figure 4. resolution 3000x4500 and in orthographic camera with $\alpha=0.0005$

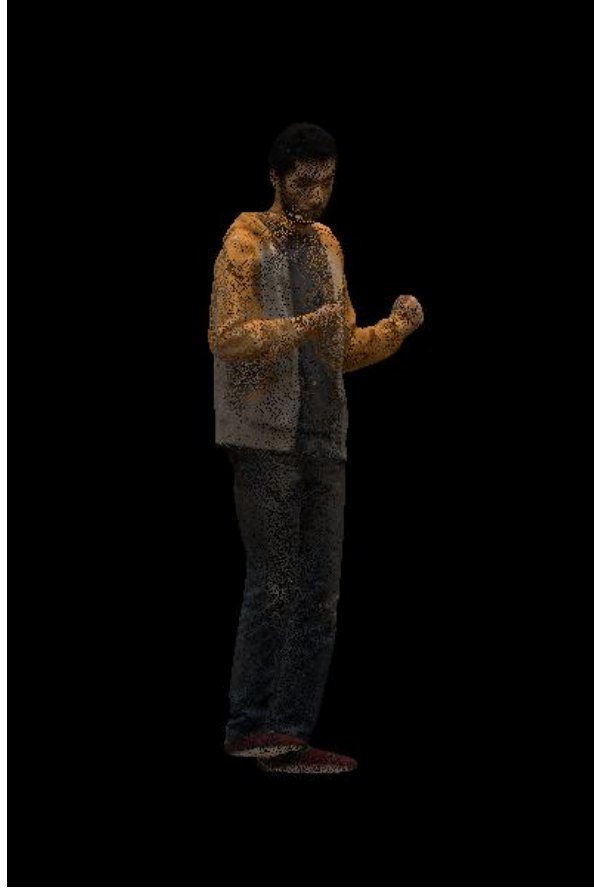


Figure 5. resolution 400x600 and in perspective camera with a focal length of 0.5, $\alpha = 0.0005$



Figure 6. resolution 500x500 and in orthographic camera with $\alpha = 0.005$



Figure 7. resolution 500x500 and in orthographic camera with $\alpha=0.005$, beta = 180