Project I: Dolly Zoom

1 Compute Focal Length and Position

1.1 Introduction

In this project, we will implement the Dolly Zoom effect used by film-makers to create a sensation of vertigo, a "falling-away-from-oneself feeling". This project is fairly simple and is meant to introduce you to the concepts of projection and focal length. The Dolly Zoom keeps the size of the object of interest constant in the image, while making the foreground and background objects appear larger or smaller by adjusting the focal length and moving the camera. You will simulate the Dolly Zoom effect with a synthetic scene as shown in Figure 1, which illustrates two cubes and one pyramid seen from the top view.

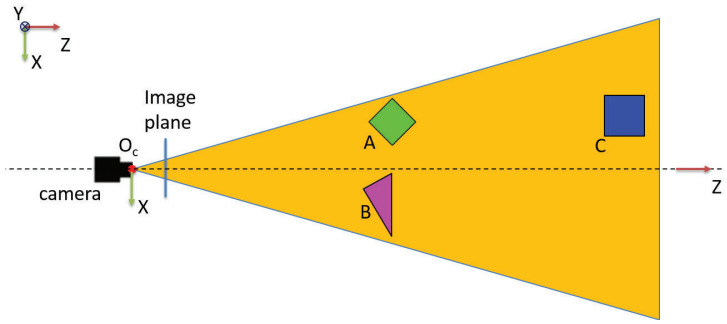

Figure 1: Dolly Zoom synthetic scene

1.2 Dolly Zoom Effect

A point in 3D is projected onto the image plane through the pinhole, or center of projection (COP):

$$u = f\frac{X}{Z}, v = f\frac{Y}{Z}$$

where (u, v) is the image coordinate of the projection, (X, Y, Z) is the 3D point, and f is the focal length of the camera. When the camera moves along the Z-axis, the depth Z changes and therefore, the projection (u, v) changes. In our particular case, the depth Z of interest is d_{ref}, the depth of the objects in the scene. In the following discussion we will only mention the u coordinate to simplify the equations, as we are focused mainly on height for the Dolly Zoom.

This projection change produced by the depth change can be compensated by adjusting the focal length:

$$u = f_{ref}\frac{X}{d_{ref}} = f'\frac{X}{d_{ref} - pos}$$

where *pos* is the movement of the camera along its Z axis (+ direction of *pos* indicates approaching to objects as shown in Figure 2) and f' is the modified focal length. f_{ref} and d_{ref} are the focal length and depth of an object in the original image, respectively. The Dolly Zoom effect exploits the compensation between depth and focal length, which produces the vertigo sensation. The relationship between all the variable names as given in the code is described in Figure 2, and when implementing the description in the code you should reference it.

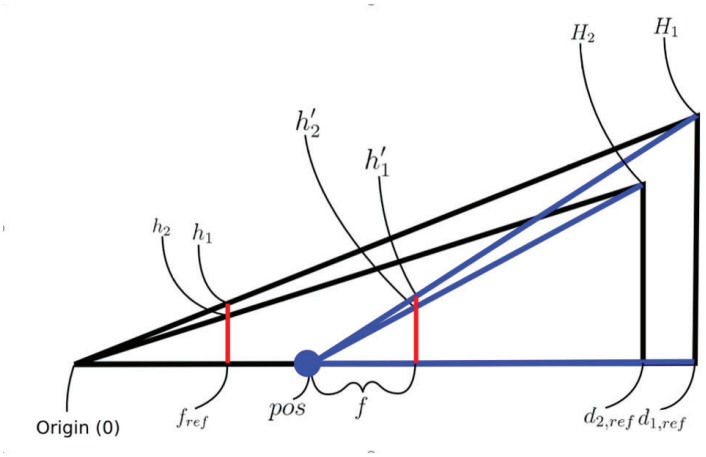

Figure 2: Relationship between variable names in the code

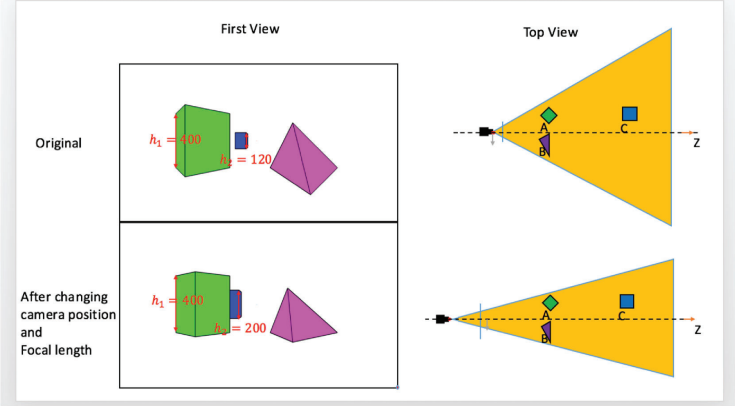

Figure 3: Focal Length/Depth Compensation

Finally, Figure 3 illustrates the focal length/depth compensation: the camera moves away from the object while changing its focal length such that the height of the object A, $h_1 = 400$, in both original and moved images remains constant. Note that the heights of the other background objects are changed due to this effect.

1.3 Implementation

For the first part of this project, you need to implement the function **compute_focal_length**. Given the depth of the object of interest d_{ref}, and the focal length f_{ref} of the camera for the original image, you need to estimate the modified focal length f', such that the height of the object remains constant as the camera moves in the Z-axis (different input pos values).

2 Compute focal length

For the second part of this project, you need to implement the function **compute_f_pos**. This time you are given the distances of two objects ($d_{1,ref}$ and $d_{2,ref}$) and their corresponding heights in the physical world (H_1 and H_2 respectively), and for a specified original focal length f_{ref}, you need to estimate the modified focal length f' such that the ratio of the heights of the two objects in the image h'_1/h'_2 is the same as the input value $ratio$, while the height h'_1 of the first object remains constant. The description and the notation of the problem is the same as the one presented in the previous part, so you can use it as reference.