Assignment2 for COMP776, ”Computer Vision”, Fall 2015

Dong Nie

1) Dolly Zoom Problem

The problem provides us an image, and a dense point cloud augmented with RGB color information, and asks us to capture a sequence of frames with “dolly zoom” effect, also we have to render it to approximate photorealistic effect. To solve this problem, there are three key points.
At first, to make the object be scaled to a specific area (400, 640) on the camera’s screen , we have to move the object (camera) to a specific position at one time. Thus, we have to compute the new depth of the foreground object. According to the optical geometry (shown in Fig.1), we can easily get compute the depth the following formulations:

$$\begin{cases} \dfrac{focal}{depth} = \dfrac{originalImageHeight}{objectHeight} \\ \dfrac{focal}{newDepth} = \dfrac{targetImageHeight}{objectHeight} \end{cases} \Rightarrow newDepth = \frac{originalImageHeight}{targetImageHeight} depth$$

Where ‘focal’ is the focal length.
Having the new depth information, we can move the object (camera) to the corresponding position at a time(The target is to make the image to be scaled to (400,640) at the first time). Then we can move the object with computed strides to capture the required frames (74 frames left).
Secondly, to capture the left frames with constant image size, we have to update the focal length for each frame. Once again, this can be completed following the simple geometry knowledge (shown in Fig.1).

$$\begin{cases} \dfrac{newFocal}{depth - stride} = \dfrac{targetImageHeight}{objectHeight} \\ \dfrac{focal}{depth} = \dfrac{targetImageHeight}{objectHeight} \end{cases} \Rightarrow newFocal = \frac{depth - stride}{depth} focal$$

Where ‘stride’ is the step length for the object(camera) to move every time.

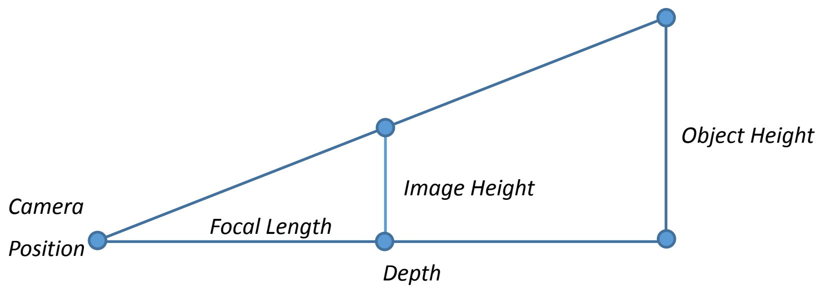

Fig.1 the optical geometry for camera/image

At last, we can use the given ‘PointCloud2Image’ function to render the frame to a 2D image.
I write the code with matlab, assuming the object moves from the initial position until most parts touching the camera screen. Then I run the program on the given dataset. The following are parts of the rendered images (I just pick one out of every 15 frames, so totally 5 images, and plus the original image in the first).

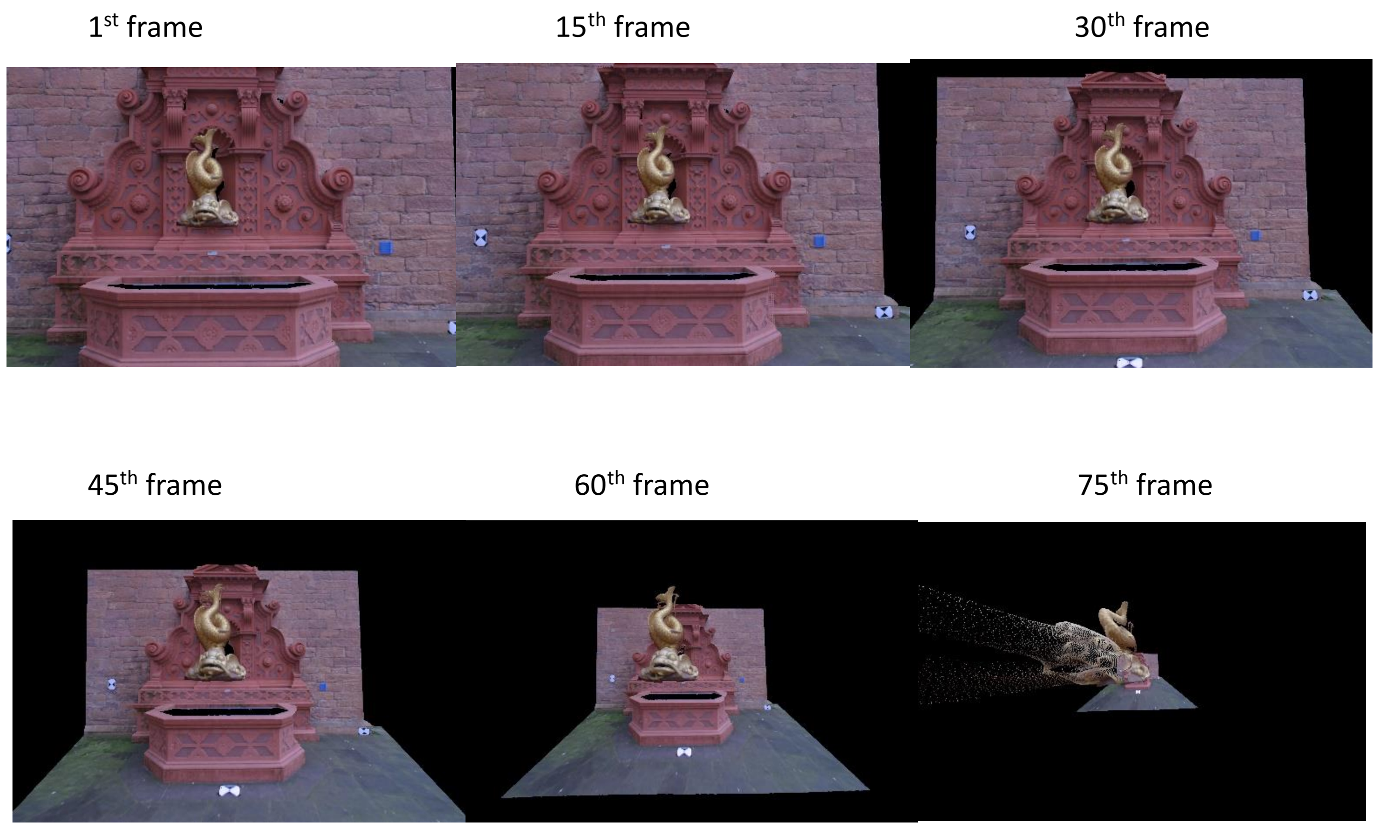

Fig.2 The dolly zoom effect for the given image

2) Radial Distortion Problem

This problem asks to distort the output images from problem1. A Brown Distortion Model is provided, however, we cannot use this model directly. To solve for a 'backwards' radial distortion model, my logic is shown in Fig.3: the key idea is build a mapping from output image coordinates to original image coordinates, and then assign corresponding pixel value to output image.

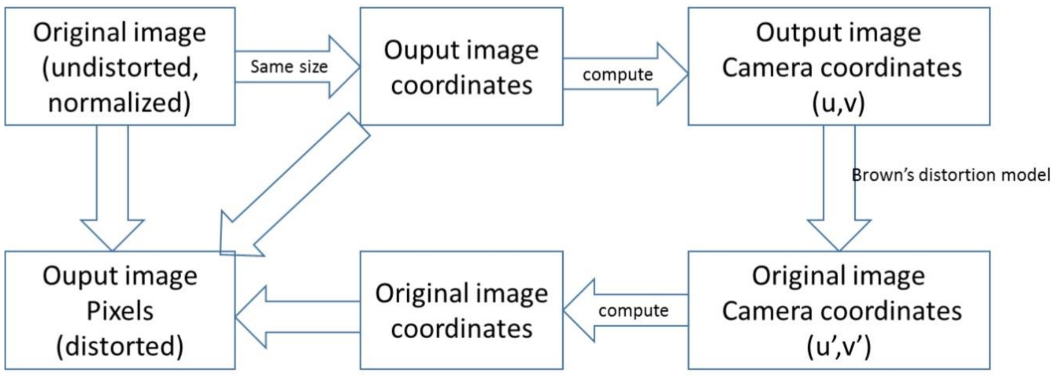
Original image (undistorted, normalized)
Same size
Ouput image coordinates
compute
Output image Camera coordinates (u,v)
Brown's distortion model
Original image Camera coordinates (u',v')
compute
Original image coordinates
Ouput image Pixels (distorted)

Fig.3 The flow chart to distort the input images

Specifically, I'll explain my solution below.
At first, build an output image according the size of input image, and normalize the coordinates of each point from the distorted output image into camera coordinates.
Secondly, for each pixel in the output image, compute the camera coordinates, and compute the corresponding camera coordinates for the input image using Brown distortion model. Also, compute the coordinates in input image according to computed camera coordinates of input image.
Thirdly, build a mapping from original input image and output image. Then for output image, we can copy the RGB value from the corresponding pixel in the original image, and leave the pixel black if the corresponding pixel is out of the boundary.
In this way, the dolly zoomed image is distorted. Of course, some step information should be updated for different frames.
For a full distortion extension, we can simply set other distortion coefficients to none-zero values and use the same idea written above.
I list some sample distorted images below.

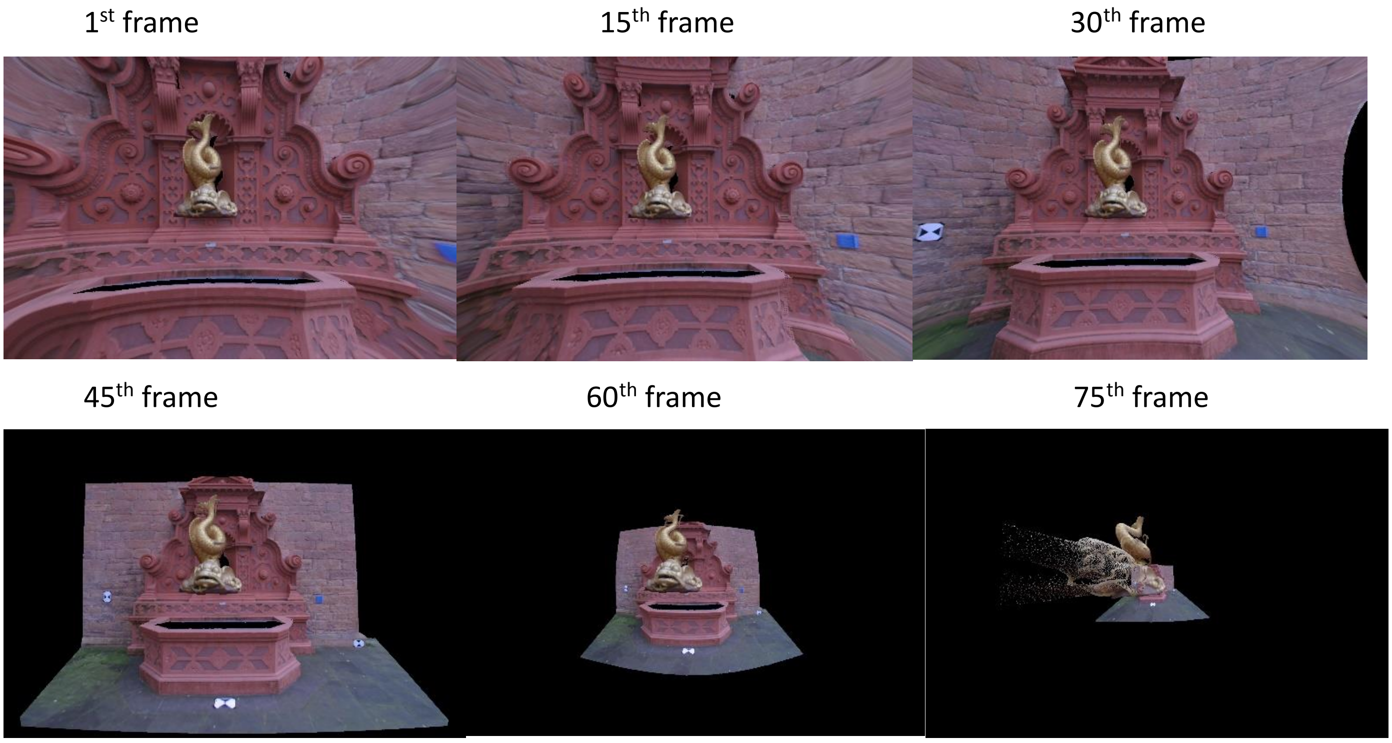

Fig.5 The distorted images for dolly zoom effect outputs

Declaration: The vedio from my program is in 'avi' format, and I convert it to 'wmv' format with the help of this link 'http://video.online-convert.com/convert-to-wmv'.