Exercise on Intrinsics, Extrinsics & Perspective-3-Point

April 30, 2019

In this exercise, we will work with a few assignments related to intrinsic and extrinsic camera parameters, camera calibration, and pose estimation using perspective-3-point method.

1 Intrinsics/Extrinsics

Here you should exercise in computing with the pinhole camera model. For you to have something to 'photograph'. a sample object is supplied in the accompanying file **Box3D.mat**. To see this box try the following script:

```
clear
close all
Q=Box3D;
plot3(Q(1,:),Q(2,:),Q(3,:),'.'),
axis equal
axis([-1 1 -1 1 -1 5])
xlabel('x')
ylabel('y')
zlabel('z')
```

Q1: The intrinsics and extrinsics are given as:

- the rotation is given by the function $\mathbf{R}$ =**Rxyz**(0.2,-0.3,0.1).
- $\mathbf{t} = \begin{bmatrix} 0.88 \\ 0.57 \\ 0.19 \end{bmatrix}$.
- $f_x = f_y = 1000,\ p_x = 300, p_y = 200.$

Form the projection matrix as $\mathbf{P} = \mathbf{K}\,(\mathbf{R}|\mathbf{t})$ and project the 3D points from the Box3D as

$$\mathbf{x} \sim \mathbf{PX} = \mathbf{K}\,(\mathbf{R}|\mathbf{t})\,\mathbf{X}$$

where $\sim$ means there is an homogeneous-to-inhomogeneous transformation. Plot the image you get.

Q2: Using the same pinhole camera as in Question 1, extend the model with radial distortion with $a_1 = -5e^{-1},\ a_2 = -3e^{-1},\ a_3 = -5e^{-2}$ and the radial distortion model as follows:

- apply extrinsics
$$\mathbf{x}' = (\mathbf{R}|\mathbf{t})\,\mathbf{X}$$
- to inhomogeneous coordinates
$$\mathbf{x}''(1) = \frac{\mathbf{x}'(1)}{\mathbf{x}'(3)},\ \mathbf{x}''(2) = \frac{\mathbf{x}'(2)}{\mathbf{x}'(3)}$$
- apply radial distortion model
$$\begin{aligned} r^2 &= \mathbf{x}''(1)^2 + \mathbf{x}''(2)^2 \\ \mathbf{x}''(1) &= \mathbf{x}''(1)(1 + a_1 r^2 + a_2 r^4 + a_3 r^6) \\ \mathbf{x}''(2) &= \mathbf{x}''(2)(1 + a_1 r^2 + a_2 r^4 + a_3 r^6) \end{aligned}$$

- multiply $\mathbf{K}$

$$\mathbf{x}''' = \mathbf{K} \begin{bmatrix} \mathbf{x}''(1) \\ \mathbf{x}''(2) \\ 1 \end{bmatrix}$$

project the Box3D points and compare to the results from Question 1. Plot again and observe what has changed.

2 Camera Calibration

In this exercise, you are asked to do the camera calibration for given images. There are two calibration toolboxes that you can use:

- **Camera Calibrator** from MATLAB computer vision toolbox.
- **Camera Calibration Toolbox** for MATLAB by Jean-Yves Bouguet[1]: this toolbox is one of the earliest camera calibration toolbox.

Q3:

1. read relevant docs to learn how to use those toolboxes.
2. load images prepared in the sub-folder **calibration** and complete camera calibration, the chessboard (or called checkerboard) square size is 112 mm.
3. verify the calibrated results and interpret the results you get.

Q4: Use the previous calibrated parameters to rectify a distorted image **distort.bmp**. **Hints:** Recall that the distortion models are nonlinear, so it is not trivial to directly find the undistorted points from distorted points with known distorsion parameters. In stead, we can do it as follows:

[1] http://www.vision.caltech.edu/bouguetj/calib_doc/

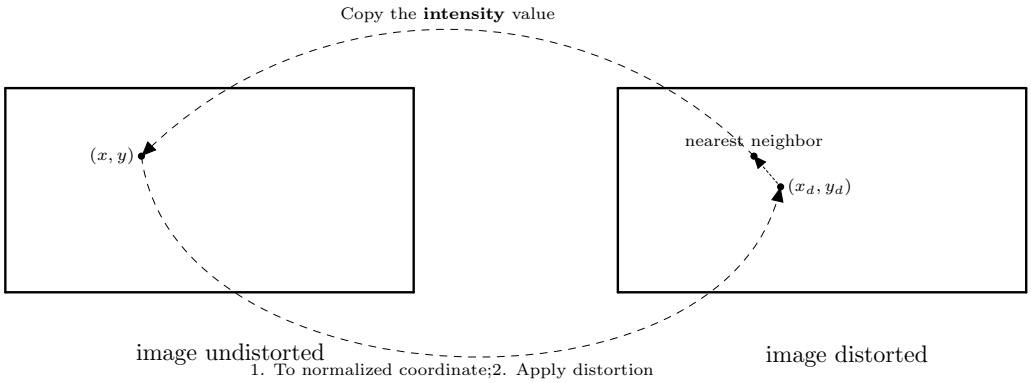

Figure 1: Illustration of the rectification process: the objective is to recover an undistorted image from a distorted image. Imaging we have an undistorted image, for every pixel (x, y), if we apply the distortions, we will get its distorted position (x_d, y_d) in the distorted image. Theoretically, we just need to copy the intensity of (x_d, y_d) as the intensity of (x, y). However, (x_d, y_d) may be float values which doesn't correspond to any integer pixels. Here we can simply choose the nearest neighbour and copy its intensity to the intensity of (x, y). We do this for every pixel of the undistorted image. Note if one pixel, after applying distortion, is not within the distorted image, we simply skip this pixel and leave its intensity unchanged.

- Initialize a blank image.
- For each pixel (x, y), compute the so called normalized coordiante (x', y') as $x' = \frac{x-p_x}{f_x},\ y' = \frac{y-p_y}{f_y}$, p_x, p_y **are coordinate for principle point**.
- Apply the radial and tangent distorsions to (x', y') and denote the result as (x_d, y_d) as

$$\begin{aligned} r^2 &= x'^2 + y'^2 \\ x'' &= x'(1 + a_1 r^2 + a_2 r^4 + a_3 r^6) + (2p_1 x' y' + p_2(r^2 + 2x'^2)) \\ y'' &= y'(1 + a_1 r^2 + a_2 r^4 + a_3 r^6) + (p_1(r^2 + 2y'^2) + 2p_2 x' y') \\ x_d &= f_x x'' + p_x \\ y_d &= f_y y'' + p_y \end{aligned}$$

  here, a_1, a_2, a_3 **are radial distortion coefficients and** p_1, p_2 **are tangent distortion coefficients**.
- Find the nearest neighbor of (x_d, y_d) in distorted image and copy the corresponding intensity (or RGB) for (x, y).

Better results can be obtained by applying bilinear interpolation.

3 Perspective 3 Point

Load the **p3p.mat** which contains:

- $\mathbf{K}$: simulated camera intrinsics;
- $\mathbf{R}_i,\ \mathbf{t}_i,\ i = 1, 2, 3$: ground truth extrinsics;
- $\mathbf{x}_i,\ \mathbf{X}_i,\ i = 1, 2, 3$: $\mathbf{X}_i$ denotes raw 3D point and $\mathbf{x}_i$ represents the corresponding image point of $\mathbf{X}_i$, i.e.

$$\mathbf{x}_i \sim \mathbf{K}(\mathbf{R}_i \mathbf{X}_i + \mathbf{t}_i)$$

The Perspective 3 Point can be decoupled into 2 steps

1. Recover the 3D points in camera coordinate system (recall the lectures on cosine rule, fourth order polynomial, etc). **Hints:**
    1. choose three points from p_1 and q_1, e.g.
2. Use provided code **lssol.m** to recover the extrinsics.

Figure 2: Example of undistorted and corresponding rectified image.

You should use 3 points for the Perspective 3 Point algorithm. Since multiple solutions may be obtained, you need at least a fourth point to choose the correct one. If you have more then 4 points, you can select the best one by computing the re-projection errors for all points and select the one with the minimum re-projection error:

$$\text{Apply extrinsics: } \mathbf{x}_i' = \mathbf{K}(\mathbf{R}\mathbf{X}_i + \mathbf{t}),$$

$$\text{Homogeneous to inhomogeneous: } \mathbf{x}_i' = \frac{\mathbf{x}_i'}{\mathbf{x}_i'(3)}$$

$$\text{Compute reprojection error: } \mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R},\mathbf{t}}{argmin} \sum_{i=1}^{n} ||\mathbf{x}_i' - \mathbf{x}_i||$$

You can check your recovered results using the following code:

```
Cc1 = -Rc1'*tc1;% recover camera center
Cc2 = -Rc2'*tc2;% recover camera center
Cc3 = -Rc3'*tc3;% recover camera center

C1 = -R1'*t1;% recover camera center
C2 = -R2'*t2;% recover camera center
C3 = -R3'*t3;% recover camera center

figure
cam1 = plotCamera('Location',C1,'Orientation',R1,'Opacity',0.0,'Color',[1 0 ...
    0],'Label','Camera1');hold on;
cam2 = plotCamera('Location',C2,'Orientation',R2,'Opacity',0.0,'Color',[1 0 ...
    0],'Label','Camera2');
cam3 = plotCamera('Location',C3,'Orientation',R3,'Opacity',0.0,'Color',[1 0 ...
    0],'Label','Camera3');

camc1 = plotCamera('Location',Cc1,'Orientation',Rc1,'Opacity',0.2,'Color',[0 1 ...
    0],'Size',0.5,'Label','Est1');
camc2 = plotCamera('Location',Cc2,'Orientation',Rc2,'Opacity',0.2,'Color',[0 1 ...
    0],'Size',0.5,'Label','Est2');
camc3 = plotCamera('Location',Cc3,'Orientation',Rc3,'Opacity',0.2,'Color',[0 1 ...
    0],'Size',0.5,'Label','Est3');
xlabel('x:(m)','FontName','Aerial','FontSize',15);
ylabel('y:(m)','FontName','Aerial','FontSize',15);
zlabel('z:(m)','FontName','Aerial','FontSize',15);
```

Result obtained:

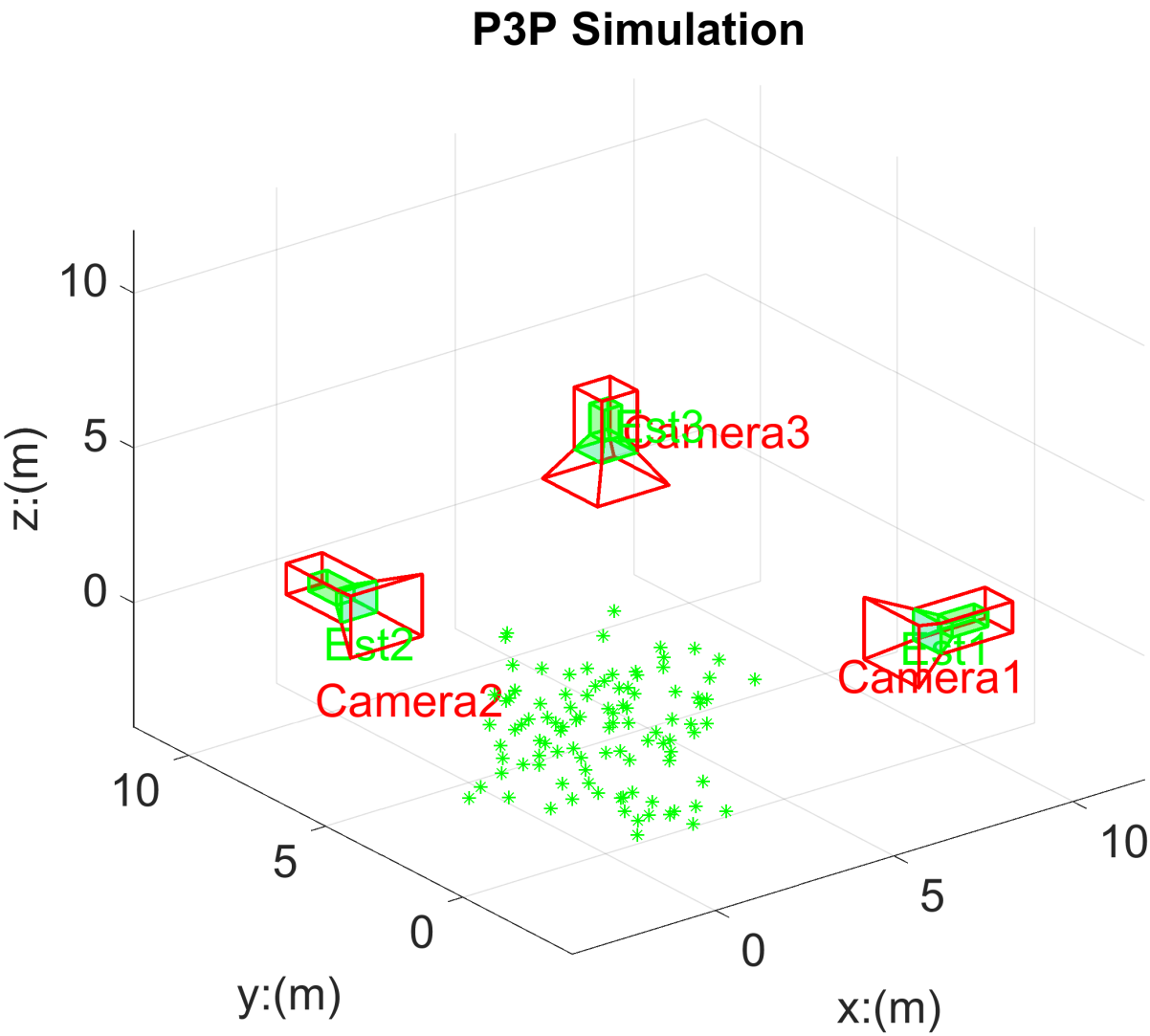

Figure 3: Result obtained by applying Perspective-3-Point Method.