Part1：

Calibration process:

Step1：

we use zhang’s method to get all parameter about the camera, which is shown on https://research.microsoft.com/en-us/um/people/zhang/calib/.

In our case, the camera parameter is as follows:

```
% Intrinsic and Extrinsic Camera Parameters
%
% This script file can be directly excecuted under Matlab to recover the camera intrinsic and extrinsic parameters.
% IMPORTANT: This file contains neither the structure of the calibration objects nor the image coordinates of the calibration points.
%          All those complementary variables are saved in the complete matlab data file Calib_Results.mat.
% For more information regarding the calibration model visit http://www.vision.caltech.edu/bouguetj/calib_doc/


%-- Focal length:
fc = [ 2904.270361101155700 ; 2898.105964593801200 ];

%-- Principal point:
cc = [ 1930.031305256202800 ; 1336.398112135144900 ];

%-- Skew coefficient:
alpha_c = 0.000000000000000;

%-- Distortion coefficients:
kc = [ -0.102803787145422 ; 0.025873118698851 ; 0.001180646763699 ; 0.001212172123117 ; 0.000000000000000 ];

%-- Focal length uncertainty:
fc_error = [ 6.673061754506680 ; 6.828562679291992 ];

%-- Principal point uncertainty:
cc_error = [ 4.538799608026081 ; 5.004338411632729 ];

%-- Skew coefficient uncertainty:
alpha_c_error = 0.000000000000000;

%-- Distortion coefficients uncertainty:
kc_error = [ 0.005140596913078 ; 0.019731781282229 ; 0.000436766900957 ; 0.000422821522463 ; 0.000000000000000 ];
```

```
%-- Image size:
nx = 3872;
ny = 2592;
```

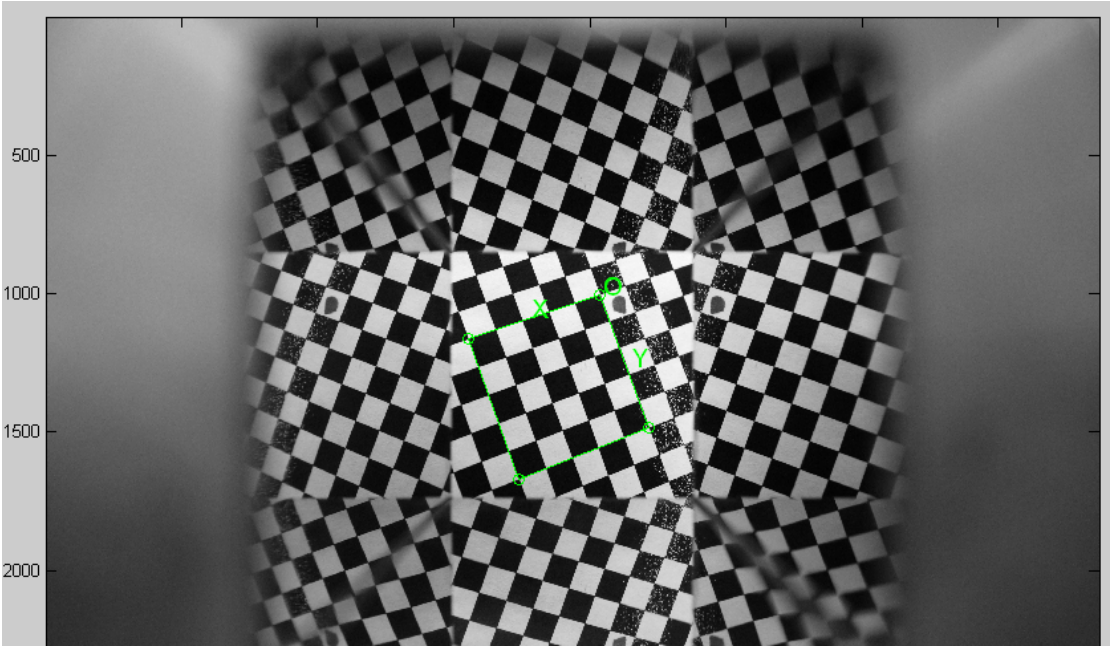
Step1

Choose a picture from our dataset. And then we extract a region in the middle part of this image like the follow image shows. We can compute the translation vector and rotation matrix by which we can change the camera coordinate system to the new coordinate system based on the chessboard in middle part of the image.

Translation vector: T1 = [5.975152 -18.615119 160.505877]

Rotation matrix: R1= [-0.909052 0.347239 0.230324
0.330610 0.937506 -0.108531
-0.253616 -0.022513 -0.967043]

Pixel error: err = [0.38415 0.32308]



Translation vector: T1 = [5.975152 -18.615119 160.505877]

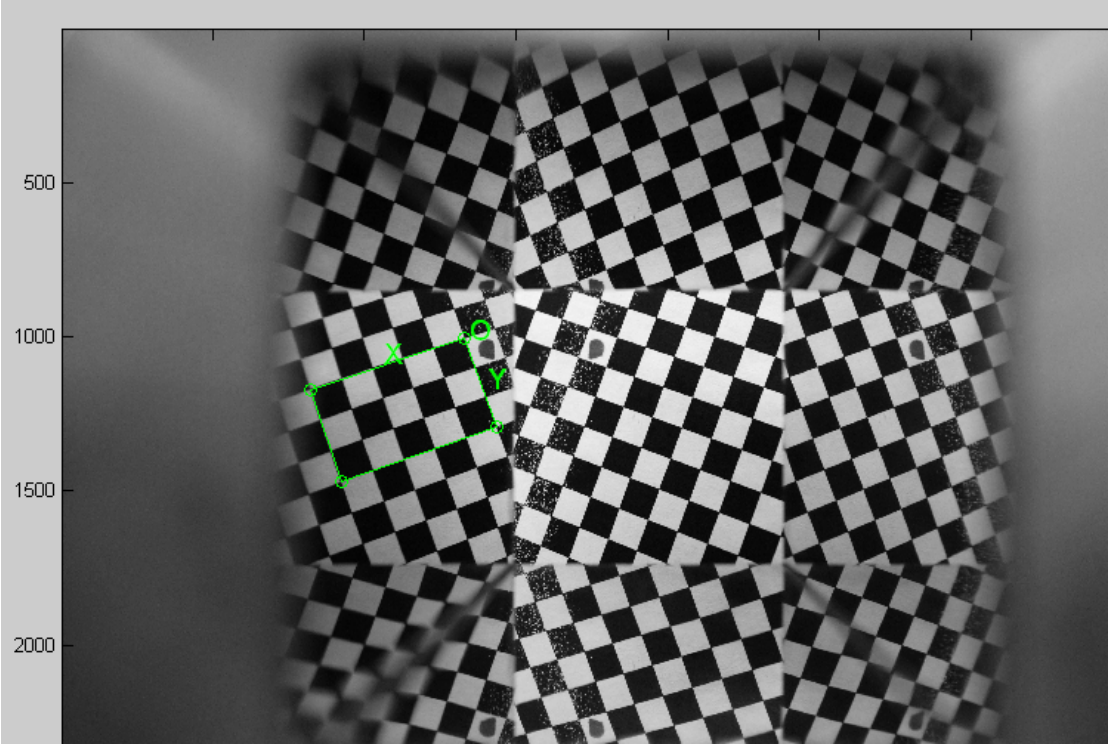
Rotation matrix: R1= [-0.909052 0.347239 0.230324
0.330610 0.937506 -0.108531
-0.253616 -0.022513 -0.967043]

Pixel error: err = [0.38415 0.32308]

Step3:

Since the virtual camera has a different CCD system from the normal camera, we need to flip the whole image from left to right, that is equal to flip the CCD system in order to get the normal one.

And then we choose the corresponding part on the left image and right image and calculate the same information concerning transformation matrix and rotation matrix.



Extrinsic parameters:

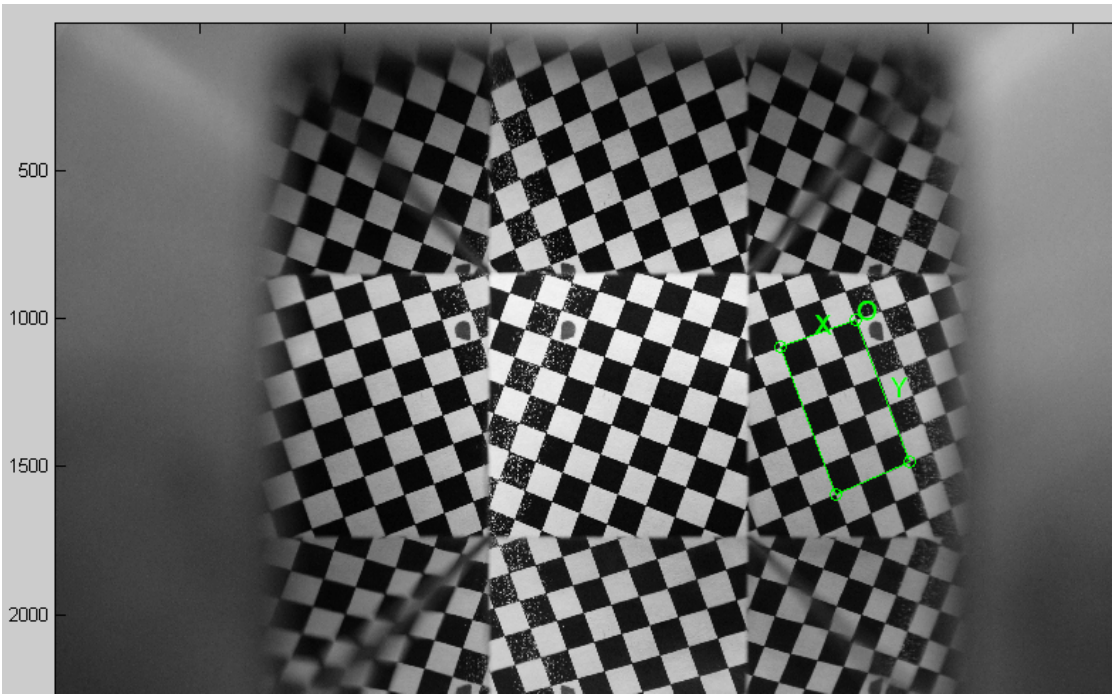
Translation vector: T2 = [-33.049912 -18.422844 158.377038]

Rotation matrix: R2= [-0.914710 0.355965 0.191296

0.331898 0.931807 -0.146896

-0.230541 -0.070877 -0.970478]

Pixel error: err = [1.21869 0.65986]



Extrinsic parameters:

Translation vector: T3= [45.753953 -18.525502 159.470729]
Rotation matrix: R3 = [-0.908789 0.348160 0.229972
0.326651 0.936567 -0.127051
-0.259618 -0.040342 -0.964868]
Pixel error: err = [1.00048 0.35242]

Then T3-T1 and T2-T1 will be the virtual camera center's positions.
And inv(R2)*R1 and inv(R3)*R1 will be the rotation matrix.

Part2：
Tooth depth result:

I use a same way that deal with the can picture to deal with a colored tooth. This time, I think we get some reasonable results. We can see it from the points set in 3D scene.

Original object:

Positive things:

1 the whole shape is very like the original model.

2 most surfaces' points are continuous.

According to this, we may produce a good result about the teeth reconstruction in future.

But we still have some work need to do:

1 reduce noise: it is still produce several points that should not belong to the model

2 modify our image system: we have 5 images, but I can only use three of them. Suppose the tub's center is (x,y)， we can rotate the camera and tub to make the camera's z-axis through x. But we can't move the camera up and down to make it through y. So the images on vertical direction do not supply any help now. In corresponding points finding part, a special case often happens that there are two or more points on horizontal line, so we may get the same wrong answers in both left and right images. However, this case can't happen on both vertical direction and horizontal direction. If we can make use of the vertical direction's image, we can tell which corresponding points are right. It will be really helpful.

3 Add adjust parameters to our depth equation, we are now try to get it from the virtual camera.


Displaying


Displaying


Displaying