# 3D Object reconstruction by camera calibration and coordinate transformations

**Rahul Lele**
**Silu Yang**

CS 211A (M. Gopi) Homework 2 – Fall 2013

**Purpose**

To learn the methods behind camera calibration, matrix decomposition, and coordinate transformations with the goal of reconstructing a 3D object by probing points on the object's surface. This write-up will walk through these stages and will comment on the issues and successes we encountered.

**Overview of the process**

The essence of this assignment is to take a picture of a setup depicting a 3D global coordinate system (Figure 1), and calculating the 3D position and orientation of the camera with respect to the origin of the global coordinate system. By determining this transformation, one can then calculate the position of the tip of the attached probe with respect to the global coordinate system. By probing multiple points of a 3D object (using a pen attached to a camera), we can then collect these points and plot them as a reconstruction of that object.

The three main steps required to do this are calibration of the camera (which results in a camera calibration matrix), decomposition of the obtained camera calibration matrix, and the transformation of the probe's camera coordinates to global coordinates. The camera calibration step produces a matrix that completely describes the projection of 3D global coordinates to pixels in an image. The matrix decomposition step results in matrices that describe the camera's position and orientation with respect to the global origin. Lastly, the decomposed matrices are used to determine the global coordinates of the probe.

**Pinhole camera model**

Cameras project the 3D world onto 2D images. The pinhole camera model is a simple description of how 3D world points are mapped onto 2D images. This model is detailed in the course notes [1], and is the model we use in this assignment.

This model states that

$$P_c = CP \tag{1}$$

or alternatively written,

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = C \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2}$$

where $P$ is a homogeneous coordinate representing a point in 3D (4D homogeneous), $C$ is the complete camera calibration matrix (3 x 4 matrix), and $P_c$ is the projection of the 3D point $P$ onto a plane parallel to the camera's image plane. Dividing the homogeneous coordinates of $P_c$ by $w$ give the image pixel coordinates of the projected 3D point.

In other words, the matrix $C$ encapsulates how 3D points are mapped onto 2D images. Once the camera calibration matrix $C$ is known, it can be decomposed into constituent matrices (which will be discussed later) that reveal much more about the system. Therefore, the first goal is to find correspondence between the known 3D points to their 2D pixel locations in the image, and to use this knowledge to solve for the camera calibration matrix.

**Step 1A - Camera calibration: automated 3D to 2D point correspondence**

We chose to automate our point correspondence, as this allowed us to take many pictures, and simply batch process them to get an outputted reconstruction. This also gave us the ability to test and revise our code on the fly since new feedback was instantly available. Figure 1 shows the global coordinate system.

**Figure 1.** The global coordinate system. Twelve points are used to calibrate the camera – three of each color of RGBK. The *x* axis points up, the *y* axis points right, and the *z* axis points into the wall. The leftmost green dot is (0,0,0).

The method we devised to locate the pixel coordinates of the points is based on thresholding the RGB values of the pixels and tagging them (which is why we used red, green, "blue", and black dots). Once the pixels are tagged with their respective colors, the centers of masses of the tagged pixels give the (x,y) pixel coordinates of the points.

There is still the issue of determining the coordinates of the three points in each tagged image – the global coordinates of the tagged points could be any of three possible points. The actual correspondence algorithm used is unique to this setup. With four colors, six combinations of two colors can be made. With this in mind, the points were placed in pairs all over the coordinate system. The premise is that you can scan for pairs of points in the sum of two tagged image masks, and two points of opposite color that are closest to each other correspond to the known global coordinates of the pair. Figure 2 depicts the success of this correspondence algorithm.

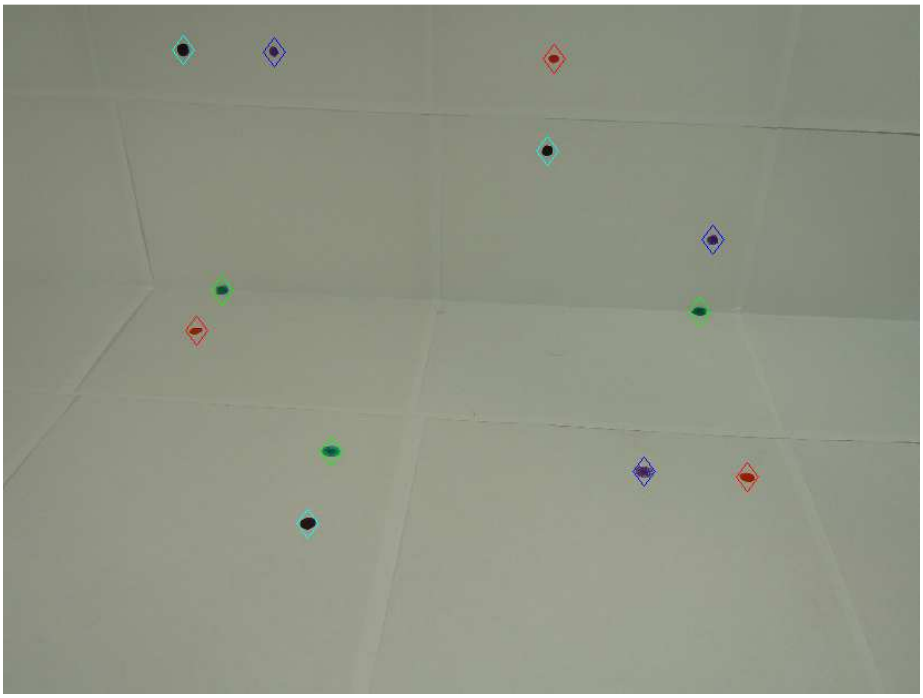


**Figure 2.** Point correspondence algorithm based on RGB thresholding. Pixels are tagged according to their color. With four colors, six pairs of points can be placed. Our correspondence algorithm works by scanning for these pairs (as described above).

**Step 1B - Camera calibration: solving for the calibration matrix**

Once the point correspondences are obtained, the next step is to solve for the camera calibration matrix. From equation 2 for a particular point, $P_1$, the equation becomes

$$\begin{pmatrix} u_1 \\ v_1 \\ w_1 \end{pmatrix} = C \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{pmatrix} \tag{3}$$

As mentioned above, the pixel coordinates arise from this equation when dividing $u_1$ and $v_1$ by $w_1$. If $C$ can be denoted as

$$C = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \tag{4}$$

then the point correspondences produce the equations

$$u_1' = \frac{u_1}{w_1} = \frac{r_1 \cdot P_1}{r_3 \cdot P_1} \tag{5}$$

$$v_1' = \frac{v_1}{w_1} = \frac{r_2 \cdot P_1}{r_3 \cdot P_1} \tag{6}$$

These equations can be rearranged as

$$u_1'(r_3 \cdot P_1) - r_1 \cdot P_1 = 0 \quad (7)$$

$$v_1'(r_3 \cdot P_1) - r_2 \cdot P_1 = 0 \quad (8)$$

As shown, each correspondence produces two linear equations relating global coordinates and pixel locations. Since the calibration matrix *C* has 12 unknowns, at least six points (12 equations) are needed to solve the system for a best fit. With the algorithm described in the previous section, we obtain 12 correspondences resulting in 24 equations. The system that arises is of the form

$$Ax = 0 \quad (9)$$

In our case, since this is an over-constrained system (and because the trivial solution is not helpful), singular value decomposition is used to solve for the best-fit vector *x* that best satisfies this system (this process is carried out in scripts main_03 and main_04). To test the accuracy of the calibration matrix, we compared the actual pixel coordinates to those predicted by the camera calibration matrix. Table 1 shows this comparison for one image.

**Table 1.** Sample comparison (ratios) of actual pixel coordinates vs. those predicted by matrix *C*.

| | ***x*** | ***y*** |
|---|---|---|
| **P1** | 1.0005 | 1.0019 |
| **P2** | 1.0002 | 0.9984 |
| **P3** | 0.999 | 0.9997 |
| **P4** | 0.9997 | 0.9999 |
| **P5** | 1.0014 | 1.0011 |
| **P6** | 0.9972 | 0.9994 |
| **P7** | 1.0016 | 0.9988 |
| **P8** | 0.9988 | 1.0029 |
| **P9** | 1.0005 | 0.9988 |
| **P10** | 0.9923 | 1.0019 |
| **P11** | 1.0059 | 0.9966 |
| **P12** | 1.0018 | 1.0022 |

As shown, the camera calibration matrix is very accurate.

**Step 2 - Decomposing *C* into intrinsic and extrinsic matrices**

The next step is to decompose the camera calibration matrix into its constituent matrices *K*, *R*, and *T*. *K* is the intrinsic parameter matrix, *R* is the camera rotation matrix, and *T* is the camera translation matrix. As will be discussed below, *E=(R|RT)* is known as the extrinsic parameter matrix. The following equations show the relationship among these matrices.

$$C = KE \quad (10)$$

$$C = K(R|RT) \quad (11)$$

$$C = (KR|KRT) = (\mathrm{M}|\mathrm{MT}) \quad (12)$$

Matrices *K* and *R* are the resultant matrices of the RQ decomposition of matrix *M*, and the translation vector *T* is solved for by

$$T = M^{-1}c_4 \tag{13}$$

where vector $c_4$ is the fourth column of the camera calibration matrix.

Once these matrices are obtained, some meaning can be extracted from them. *K*, the intrinsic parameter matrix, is dependent only on how the camera is designed – no matter where this camera is, this matrix will stay the same for a particular focal length (which is why it is called the *intrinsic* matrix). *E*, the extrinsic parameter matrix, is a function of the camera's location in the world with respect to the global coordinate system. *E* is not a function of the camera's design (intrinsic), but a function of the camera's location in the world (hence, it is called *extrinsic* matrix).

For this assignment, only the extrinsic matrix is needed. The extrinsic matrix defines the relationship between the global coordinate system, and the camera coordinate system – what we set out to find in the first place. With this information, points in one coordinate system are easily transformed into points in another.

**Step 3 – Coordinate transformation: calculating the global coordinates of the probe tip**

The last step before being able to probe the object, is to find the position of the probe tip with respect to the camera coordinate system. These coordinate values will not change for the duration of this assignment because the probe is fixed to the camera.

To do this, a 13$^{th}$ point was marked on the table whose coordinates with respect to the global coordinate system were known. Then, once the extrinsic matrix is found for that particular image, the extrinsic matrix is applied to the global coordinates to transform them into the camera coordinates. These new coordinates describe the probe tip with respect to the camera coordinate system. As an aside, we had to write an additional script (unify_xy) to remedy issues with sign and ordering of the camera coordinates. There are two reasons these discrepancies might have occurred. First, we didn't take into account the dimensions of the image matrix beforehand (first coordinate is vertical, second coordinate is horizontal). Also, we used the derivation of equations 7 and 8 that is in the class notes[1] without checking if our coordinate conventions matched those in the notes.

Finally, once the camera coordinates of the probe are known, finding the global coordinates of the probe for every image becomes a matter of multiplying the camera coordinates of the point by the inverse of the concatenated extrinsic matrix (concatenated with a fourth row [0 0 0 1] to make it square and invertible). Mathematically,

$$P_{probe\_gcs} = E_{square}^{-1} P_{probe\_ccs} \tag{14}$$

where "*gcs*" stands for "global coordinate system," and *ccs* stands for "camera coordinate system. Equation 14 is the crux of this assignment, and produces the numbers we are ultimately looking for.

**Probing Objects**

At this point, the scripts incorporate everything written above. Now, the only thing left to do is to probe objects. Two objects were probed – a box, and a mug. The box was chosen,

because errors in our code would be immediately visible. Since probing a box is easy, any errors we would further encounter would not because of poor probing. As shown below in Figures 3a – 3d, probing the box was a successful endeavor. Notice, however, that points are missing in the back of the box. This is because we were unable to take pictures of the global coordinate system while trying to probe backward facing facets.


**Figure 3a.** Picture of the box to be probed.

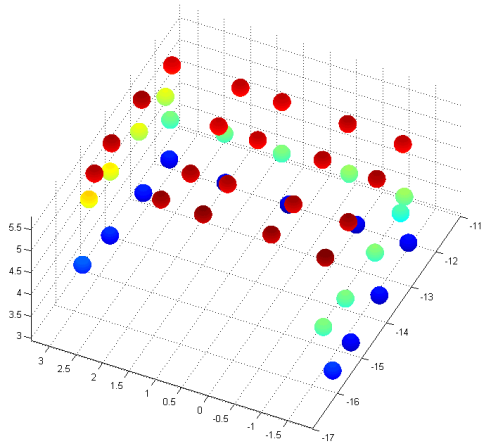


**Figure 3b.** 3D Reconstruction – same angle as picture.

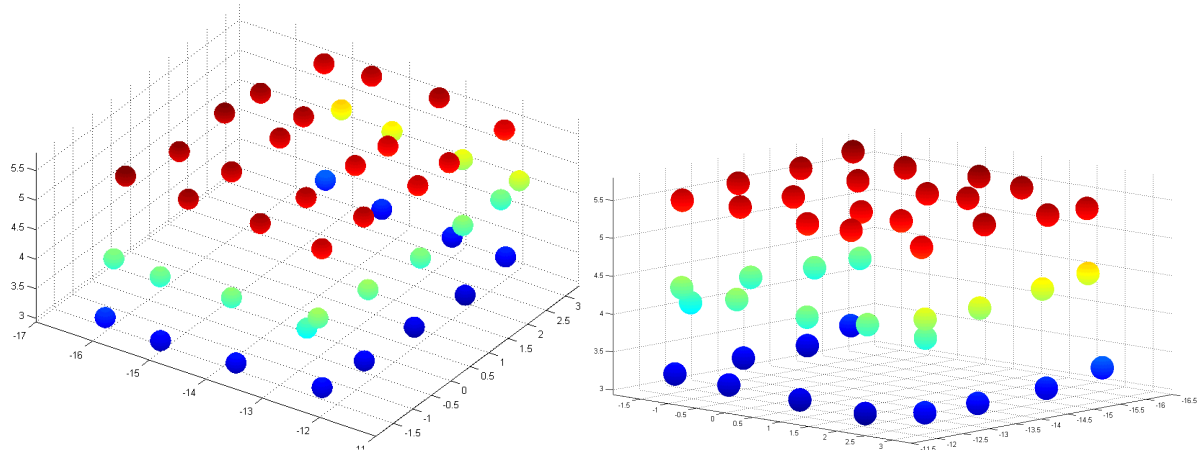


**Figure 3c – 3d.** More angles of the reconstruction. The set of images corresponding to this box is titled “box4”. To rerun the scripts, just set the variable “object_folder_name” to “box4” in the script “run_main”.

The second object, the mug, was also chosen for ease probing and discernibility of the reconstruction. The mug was a bit more difficult to probe because without probing the back side of the mug, it is hard to discern a mug from the reconstruction. After a couple tries, the points corresponding to the back side of the mug were taken from the inside of the mug. However, since the mug is not thick, this worked out. Figures 4a – 4d, depict the reconstruction of the mug from different angles. To visualize the mug more easily, we suggest running the MATLAB scripts (run_main) with the variable “object_folder_name” as ‘mug2’.

**Figure43a.** Picture of the mug to be probed.

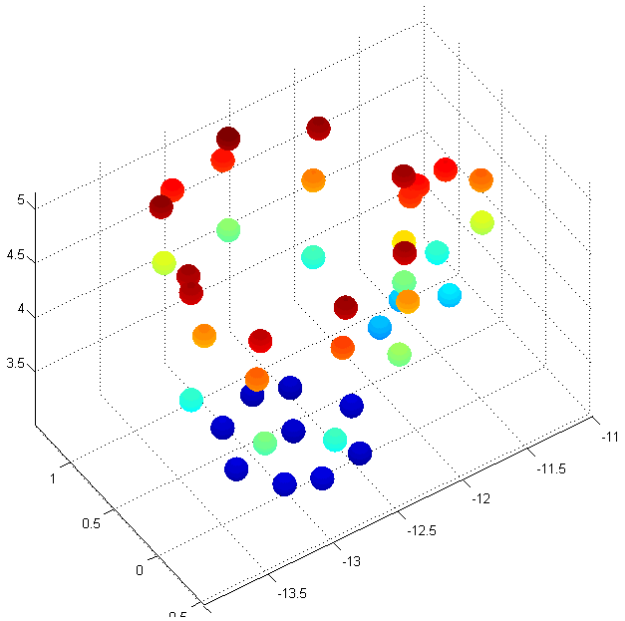
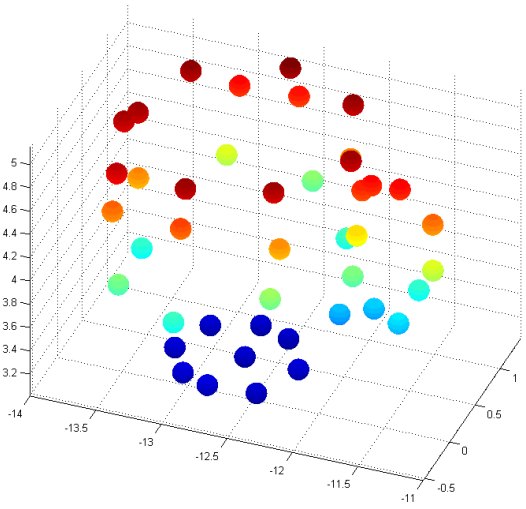



**Figure 4b.** 3D Reconstruction – same angle as picture.



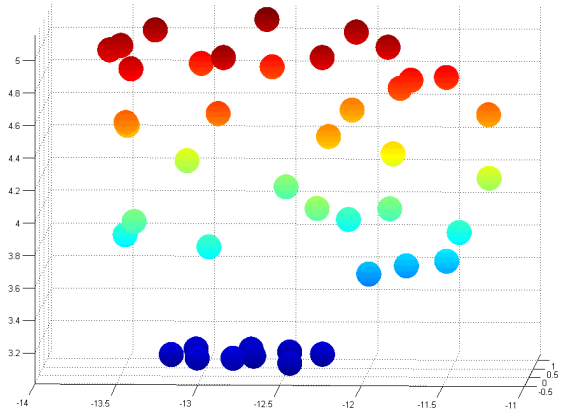




**Figure 3c – 3d.** More angles of the reconstruction. The mug is more easily discerned when rotating the figure in MATLAB. To rerun the scripts, just set the variable "object_folder_name" to "mug2" in the script "run_main".

## Conclusion

The images above are the end result of this assignment. If one were to repeat this assignment, and time permitted, it would be beneficial to create a second probe with a right angle bend to it. That way you could probe the back sides of objects.

## References

http://www.ics.uci.edu/~gopi/CS211A/classes/cameracalib.pdf
http://www.ics.uci.edu/~gopi/CS211A/assn/Assignment2.pdf