Introduction

The Perspective-n-point (PnP) problem is a very important problem in computer vision. We did not cover it in the course but this project is to let you learn this. You will need a checkerboard to provide point positions in world coordinates. You have to fix the checkerboard and the camera during steps 1-3.

Use the camera from project 3 which you know the K matrix. Once the camera the fixed, you can also obtain P, and hence recover R and t from P. Note that we use a 3D world coordinate system fixed w.r.t the checkerboard (origin at one of the corner). I assume that you have those from the previous project.

1. Please read the P3P section of this webpage:
   https://en.wikipedia.org/wiki/Perspective-n-Point
   Using the 3D points on checkerboard and the corresponding points image to solve P3P problems. The result will allow you to obtain R and t. Check if this is consistent with the result from 1.
2. Using the SolvePnP in opencv to conduct the same computation. It includes three different methods: CV_ITERATIVE, CV_P3P, and CV_EPNP. Please compare results.

The P3P (Perspective-Three-Point) Principle

The below math analysis is referenced from [1] and detailed math analysis can be found in the paper[2]. Generally speaking, the P3P principle is used to estimate the position and the orientation of an object regarding to the camera placement given the calibrated camera intrinsic parameters.

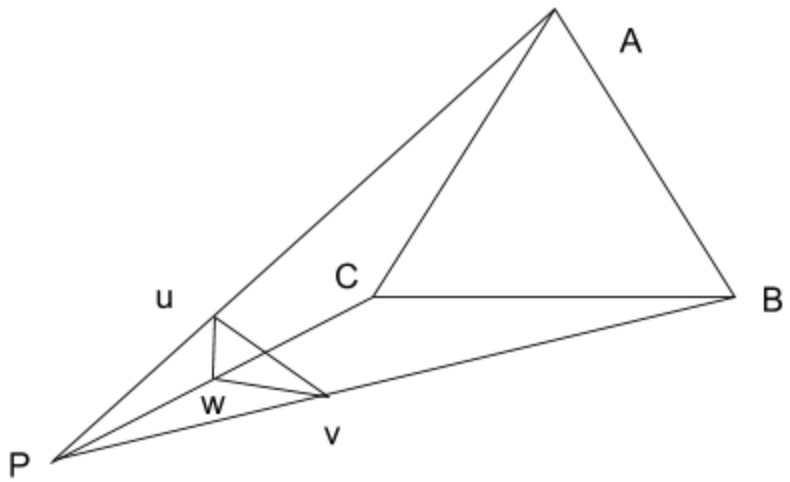
To estimate the extrinsic parameter of the camera, a few inputs are required for this homework assignments and here is the pipeline:

1. Calibrate the cameras with OpenCV to obtain the camera intrinsic parameters and find extrinsic parameters also. The intrinsic camera parameters and distortion coefficients are used as the input the P3P system.

2. Select 4 point coordinate (A,B,C,D) from the checkerboard and fix the origin to be at the top corner. The 3 points (A,B,C) will be used to find possible solutions and the 4th point D is used to verify and select best solution.
3. Find the projected checkerboard corners on the image plane, this is easily done with the OpenCV functions, undistort the image points with distortion coefficients provided in the camera calibration steps.
4. 3D points are expressed in the object coordinate system and their corresponding projections are expressed in the image coordinate system.
5. Calculate the object pose (a rotation and a translation) expressed in a coordinate system that is placed at the camera optical center, and compare the results.

The basic configuration for the P3P system is depicted below as P is the camera center, and A, B, C are the object points, u, v, w are the corresponding points on the image plane. The formulated problem here is to find the coordinate of A, B, C with respect to the coordinate system centered at the camera center P, given the image coordinates u, v, w with respect to P. One can easily find the below equation valid with very simple trigonometrical calculation, and this identity can be extended to multiple equation systems.

$$PA^2 + PB^2 - 2PA \cdot PB \cdot \cos \alpha_{uv} = AB^2$$



P3P equation system

The P3P equation system is based on the law of cosines that links affine distances with angles. Applying law of cosines as discussed above, the P3P equation system is:

$$PB^2 + PC^2 - 2.PB.PC.cos\alpha_{v,w} = BC^2$$
$$PA^2 + PC^2 - 2.PA.PCcos\alpha_{u,w} = AC^2$$
$$PA^2 + PB^2 - 2.PA.PBcos\alpha_{u,v} = AB^2$$

Some conditions have to be respected such as angles that must be comprised between 0 and π. Distances must stay strictly positive and A, B and C not co-linear. Then dividing by PC2, and introducing y = PB/PC, x = PA/PC, we obtain

$$y^2 + 1 - 2.y.cos\alpha_{v,w} - BC^2/PC^2 = 0$$
$$x^2 + 1 - 2.x.cos\alpha_{u,w} - AC^2/PC^2 = 0$$
$$x^2 + y^2 - 2.x.y.cos\alpha_{u,v} - AB^2/PC^2 = 0$$

If we let $v = AB^2/PC^2$, $av = BC^2/PC^2$ and $bv = AC^2/PC^2$. That gives:

$$y^2 + 1 - 2.y.cos\alpha_{v,w} - av = 0$$
$$y^2 + 1 - 2.x.cos\alpha_{u,w} - bv = 0$$
$$x^2 + y^2 - 2..x.ycos\alpha_{u,v} - v = 0$$

The last equation gives us the relation $v = x^2 + y^2 - 2..x.ycos\alpha_{u,v}$ Replacing v in the first two equations, we obtain the simplified version of our P3P system:

$$(1-a)y^2 - ax^2 - cos\alpha_{v,w}y + 2acos\alpha_{u,v}xy + 1 = 0$$
$$(1-b)x^2 - by^2 - cos\alpha_{u,w}x + 2bcos\alpha_{u,v}xy + 1 = 0$$

The goal of the P3P problem is to solve the P3P equation system using 3 correspondences and the values of a, b, $cos\alpha_{u,v}$, $cos\alpha_{v,w}$ and $cos\alpha_{u,w}$ Up to four set of distances $\{PA, PB, PC\}$ are obtained and used to determine the pose of the ABC triangle in the camera coordinate system (centered at P). In practical implementation, attentions has to be paid to some main steps in order to solve P3P equation system starting from 3D-2D correspondences.

Implementation Steps

The first step and very important step is to project image plane points onto the unit sphere centered on P, normalized the data, so the units from the image points can be removed.

$$u'_x = \frac{u_x - c_x}{f_x}$$
$$u'_y = \frac{u_x - c_x}{f_x}$$
$$u'_z = \frac{u_x - c_x}{f_x}$$

Here, f_x and f_y are the focal length values expressed in pixels, and c_x and c_y is the image optical center, and then we can normalized the data using the L2 norm.

$$N_u = \sqrt{(u_x^2 + u_y^2 + u_z^2)}$$
$$u''_x = \frac{u'_x}{N_u}$$
$$u''_y = \frac{u'_y}{N_u}$$
$$u''_z = \frac{u'_z}{N_u}$$

As a shorthand expression, the notation of $u'',\ v'',\ w''$ are replaced with u, v, w, then the angles can be calculated:

$$cos\alpha_{u,v} = (u_x * v_x + u_y * v_y + u_z * v_z)$$
$$cos\alpha_{u,w} = (u_x * w_x + u_y * w_y + u_z * w_z)$$
$$cos\alpha_{v,w} = (v_x * w_x + v_y * w_y + v_z * w_z)$$

As for the distance of of object in the real world, if we denote the coordinates in the real world as U, V, W respectively, the distance for each line segment can be calculated as:

$$AB = \sqrt{(V_x - U_x)^2 + (V_y - U_y)^2 + (V_z - U_z)^2}$$
$$BC = \sqrt{(W_z - V_x)^2 + (W_y - V_y)^2 + (W_z - V_z)^2}$$
$$AC = \sqrt{(W_x - U_x)^2 + (W_y - U_y)^2 + (W_z - U_z)^2}$$

In this step, we are able to calculate the coefficients a and b,

$$a = \frac{BC^2}{AB^2}$$
$$b = \frac{AC^2}{AB^2}$$

Solve the two quadratic equation is difficult, so further simplification can be made to combine the two equation into one single equation below[3], with coefficients for each variable

$$a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 0$$
$$b_1y - b_0 = 0$$

Here below is the coefficients expressed in the known variable $a,\ b,\ cos\alpha$

$$p = 2cos\alpha_{v,w},\ q = 2cos\alpha_{u,w},\ r = 2cos\alpha_{u,v}$$
$$a_4 = a^2 + b^2 - 2a - 2b + 2(1 - r^2)\,ba + 1$$
$$a_3 = -2qa^2 - rpb^2 + 4qa + (2q + pr)b + (r^2q - 2q + rp)ab - 2q$$
$$a_2 = (2 + q^2)a^2 + (p^2 + r^2 - 2)b^2 - (4 + 2q^2)a - (pqr + p^2)b - (pqr + r^2)ab + q^2 + 2$$
$$a_1 = -2qa^2 - rpb^2 + 4qa + (pr + qp^2 - 2q)b + (rp + 2q)ab - 2q$$
$$a_0 = a^2 + b^2 - 2a + (2 - p^2)b - 2ab + 1$$
$$b_1 = b\,((p^2 - pqr + r^2)a + (p^2 - r^2)b - p^2 + pqr - r^2)^2$$
$$b_0 = ((1 - a - b)x^2 + (a - 1)qx - a + b + 1)((r^3(a^2 + b^2 - 2a - 2b + (2 - r^2)ab + 1)x^3)$$
$$+ r^2(p + pa^2 - 2rqab + 2rqb - 2rq - 2pa - 2pb + pr^2b + 4rqa + qr^3ab - 2rqa^2 + 2pab + pb^2 - r^2pb^2)x^2 +$$
$$+ (r^5(b^2 - ab) - r^4pqb + r^3(q^2 - 4a - 2q^2a) + q^2a^2 + 2a^2 - 2b^2 + 2) + r^2(4pqa - 2pqab + 2pqb - 2pq - 2pqa^2) +$$
$$+ r(p^2b^2 - 2p^2b + 2p^2ab - 2p^2a + p^2 + p^2a^2))x +$$
$$(2pr^2 - 2r^3q + p^3 - 2p^2qr + pq^2r^2)a^2 + (p^3 - 2pr^2)b^2 + (4qr^3 - 4pr^2 - 2p^3 + 4p^2qr - 2pq^2r^2)a +$$
$$(-2qr^3 + pr^4 + 2p^2qr - 2p^3)b + (2p^3 + 2qr^3 - 2p^2qr)ab + pq^2r^2 - 2p^2qr + 2pr^2 + p^3 - 2r^3q$$

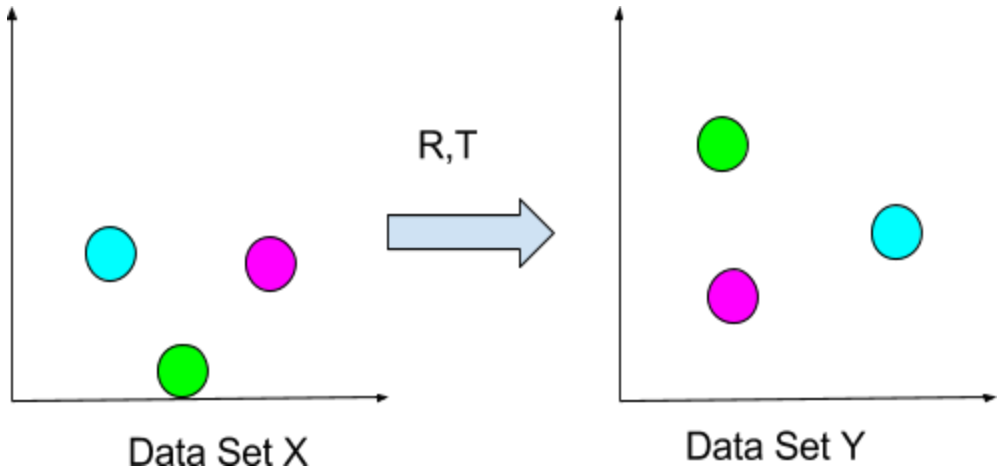
Using the “up to four” solutions we found for distances PA, PB and PC, and we can thus obtain 3D coordinates for points A, B, C by multiplying u, v, w vectors with corresponding distances,

$$A = \vec{u}\,.||PA|| \qquad B = \vec{v}\,.||PB|| \qquad C = \vec{w}\,.||PC||$$

Computing the rotation matrix and the translation vector

A, B, C 3D coordinates are expressed in the optical center coordinate system (centered at P). We wish to compute the affine transformation (R and t that transform 3D points expressed in the object coordinate system into 3D points expressed in the optical center coordinate system.

The solution to that problem is referenced from online resource[2] to find the optimal/best rotation and translation between two sets of corresponding 3D point data, so that they are aligned and registered. Here we denotes the solution coordinate(A,B,C) obtained above as Data Set X, and the coordinates in the world coordinates as Data Set Y, and we can build a translation and rotation relationship below.



The corresponding points have the same colour, R is the rotation and t is the translation. We want to find the best rotation and translation that will align the points in dataset A to dataset B. Here, ‘optimal’ or ‘best’ is in terms of least square errors. This transformation is some.

We’re solving for R,t in the equation:

$$Y = RX + t$$

Where R,t are the transforms applied to dataset X to align it with dataset Y, as best as possible. Finding the optimal rigid transformation matrix can be broken down into the following steps: find the centroids of both dataset, bring both dataset to the origin then find the optimal rotation, and find the translation t.

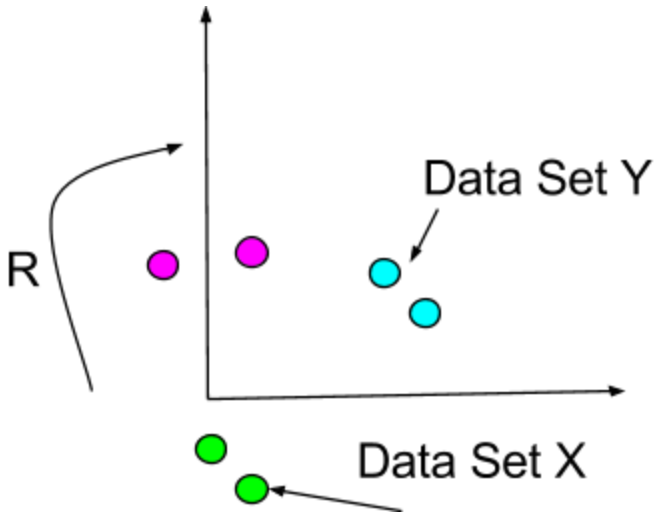
The centroid of data set X is here below assuming the points in set A and set B are represented as P:

$$centroid_X = \frac{1}{N}\sum_{i=1}^{N} P_X^i$$

$$centroid_Y = \frac{1}{N}\sum_{i=1}^{N} P_Y^i$$

Here, P_X and P_Y are points in data set X and data set Y respectively, we have to move the data to the center points at origin. In this way, the translation component can be removed and the next step involves accumulating a matrix, called E.

$$E = \sum_{i=1}^{N} (P_A^i - centroid_A)(P_B^i - centroid_B)$$



Here, I am going to use the SVD decomposition.

$$[U, S, V] = SVD(E) \quad \text{with} \quad R = VU^T$$

The special case suggested is to check the determinant of rotation matrix R to see if the determinant is -1, if so, then the 3rd column of V is multiplied by -1. The next step is to find the translation vector t and the translation t is:

$$t = -R\, centroid_X + centroid_Y$$

The centroids are 3x1 column vectors. If we look at the error terms:

$$err = \sum_{i=1}^{N} |RP_X^i + t - P_Y^i|^2$$

Plug the translation vector t into the error term leads to:

$$err = \sum_{i=1}^{N} |RP_X^i - Rcentroid_X + centroid_Y - P_Y^i|^2$$

$$err = \sum_{i=1}^{N} |R(P_X^i - centroid_X) - (centroid_Y + P_Y^i)|^2$$

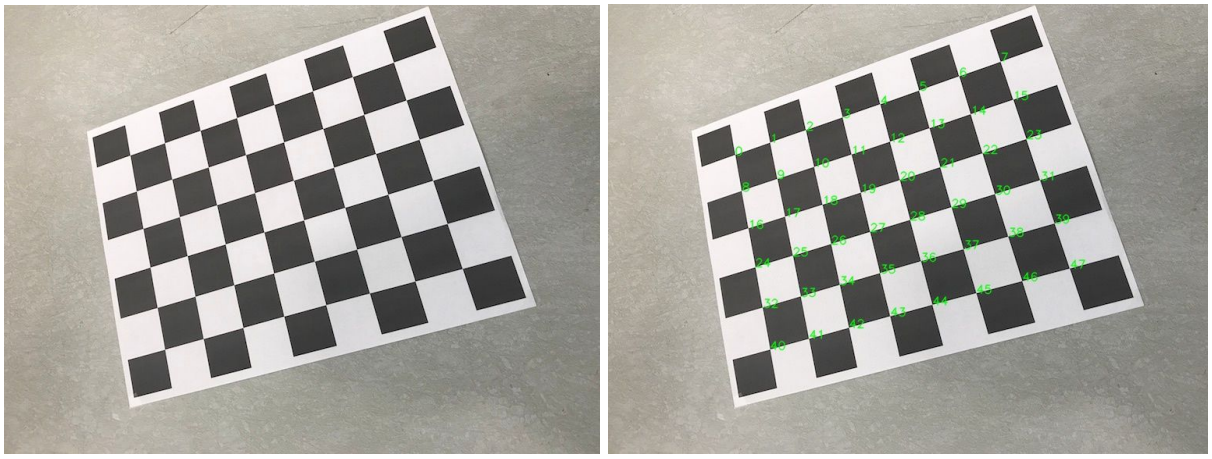
It is similar to the err term for points on the origin of the coordinate system. Four solution will be is expected for rotation and translation since the order of the equation is 4. However, we only need one valid solution instead of four and we can select the best solution to our problem by

verify the reprojection error with the fourth points D. More specifically, we can check the error term of D for each R and t, and find the least error.

$$err = \sum_{i=1}^{N} |RD_X^i + t - D_Y^i|^2$$

Result Analysis

The image I used for calculation is a standard checkerboard pattern with size (8,6), as shown in the below image.



So if you count the inner corners, you would be able to find that it has 8 corners along the horizontal line and 6 corners along the vertical line, which is shown in the figure below. In the camera calibration step, I used all these 8x6 points to calibrate the camera intrinsic and extrinsic parameters. In the world coordinate system, I fix the coordinate origin in the top corner of the image and set the third component to be 0, and then the coordinate of these 48 points in the world coordinate system is

[0, 0, 0], [30, 0, 0], [60, 0, 0], [90, 0, 0], [120, 0, 0], [150, 0, 0], [180, 0, 0], [210, 0, 0], [0, 30, 0], [30, 30, 0], [60, 30, 0], [90, 30, 0], [120, 30, 0], [150, 30, 0], [180, 30, 0], [210, 30, 0], [0, 60, 0], [30, 60, 0], [60, 60, 0], [90, 60, 0], [120, 60, 0], [150, 60, 0], [180, 60, 0], [210, 60, 0], [0, 90, 0], [30, 90, 0], [60, 90, 0], [90, 90, 0], [120, 90, 0], [150, 90, 0], [180, 90, 0], [210, 90, 0], [0, 120, 0], [30, 120, 0], [60, 120, 0], [90, 120, 0], [120, 120, 0], [150, 120, 0], [180, 120, 0], [210, 120, 0], [0, 150, 0], [30, 150, 0], [60, 150, 0], [90, 150, 0], [120, 150, 0], [150, 150, 0], [180, 150, 0], [210, 150, 0]

In the corresponding image plane, the coordinates of these 48 points are:
[162.421, 192.6], [207.086, 176.769], [252.832, 160.925], [299.817, 144.526], [348.037, 127.731], [397.668, 110.353], [449.076, 92.0778], [502.132, 73.3922], [170.559, 238.682], [216.384, 223.438], [263.361, 207.911], [311.374, 192.171], [360.414, 175.634], [411.041,

158.489], [463.716, 140.505], [518.315, 122.036], [179.278, 286.603], [226.199, 271.566], [274.074, 256.495], [323.102, 240.906], [373.089, 224.963], [424.795, 208.315], [478.815, 190.675], [534.985, 172.41], [188.381, 336.392], [236.203, 321.609], [285.223, 306.592], [335.064, 291.343], [386.312, 275.686], [439.274, 259.617], [494.743, 242.727], [552.549, 225.056], [197.51, 388.097], [246.389, 373.675], [296.304, 359.009], [347.628, 343.927], [400.124, 328.72], [454.589, 313.38], [511.73, 297.341], [571.18, 280.383], [207.258, 441.681], [256.796, 428.144], [308.021, 413.721], [360.656, 399.339], [414.67, 384.521], [470.974, 369.61], [529.748, 354.308], [590.608, 338.214]

After that, I use the OpenCV camera calibration function to calibrate the camera and return the camera intrinsic parameter, distortion coefficient, rotation as well as translation vector. Here, I have changed the representation of rotation matrix in terms of angles, with similar function `rotm2eul(rotm)` in Matlab, so the rotation matrix are expressed in terms of angles. Here below the results I obtained from Opencv camera calibration steps

Camera projection matrix P:

[1.385295946547692, 0.1703809546713676, 1.266986453652798, 164.1678436605997;
-0.5874223880353844, 1.377444481211612, 0.985151315205522, 193.3647916735719;
-0.0004219518909623706, -0.0006029479097579957, 0.002416892116503091, 1]

The camera intrinsic K is:

[646.7253043756004, 0, 372.0705710541568;
0, 651.1409268131588, 281.7410708820502;
0, 0, 1]

The distortion coefficient is D:

[0.2974056140013253, -1.547773778246966, 0.001352431982007633, 0.007509127212427604, 1.306494215048278]

The rotation matrix in calibration step is:

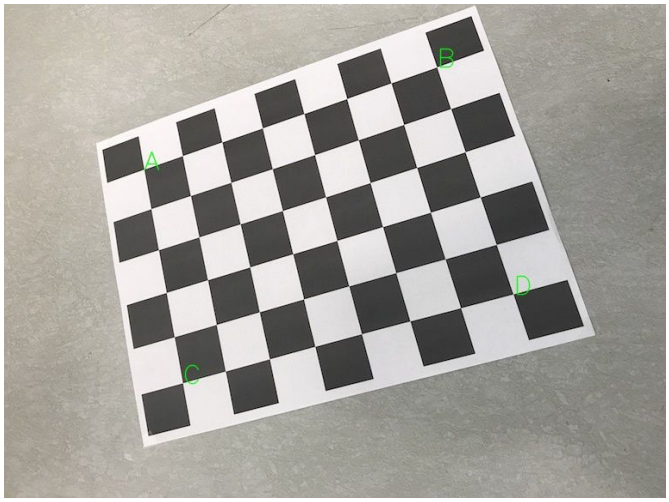
[-14.0078, 9.61423, -16.7904]

The translation vector in calibration step is :

[-127.2416729884141; -53.72170126910594; 395.8120738691324]

In the p3p step, I used the distortion coefficient and camera intrinsic parameter K as input for the p3p subroutine. Here I have selected four corner points A, B, C, D and A, B, C points are used to calculate the rotation and translation vectors. So I first undistort all the input

points with Opencv undistortion functions and then calculate the rotation and translation with the steps above.



The rotation and translation vector from p3p step is:

Rotation Matrix from p3p is :
[-12.4375, 11.1553, -16.4236]
Translation matrix from p3p is :
[-127.6017783845365; -53.80030457334301; 396.6085454255177]

The rotation and translation vector from OpenCV pnp step is:

Rotation Matrix from pnp is :
[-12.438, 11.1548, -16.4237]
Translation matrix from pnp is :
[-127.6016982321342; -53.8002946056032; 396.6082774336302]

Let's summarize the values calculated from different functions for rotation and translation components. The rotations are expressed in terms of the Angles in 3 components. It can easily be seen from the below table that the results are quite similar to each other. Among all, my implementation of P3p and OpenCV implementation Pnp has similar results. More specifically, the ***sovlepnp*** with CV_P3P has very close result due to the same math methodology used, the ***sovlepnp*** with CV_ITERATIVE has similar results with some difference in the rotation angle, and this is also same for CV_EPNP . However, the difference of rotation and translation between camera calibration and p3p can easily be justified. This is due to the fact that the math methodology differs with each other and

points used for calculation are also different. Overall speaking ,the results obtained from these 5 methods are quite similar to each other.

	Rotation 1	Rotation 2	Rotation 2	Trans 1	Trans 2	Trans 3
Calibration	-14.0078	9.61423	-16.7904	-127.2416729884141	-53.72170126910594	395.8120738691324
P3p	-12.4375	11.1553	-16.4236	-127.601778384536	-53.8003045733430	396.6085454255177
Pnp_CV_P3P	-12.438	11.1548	-16.4236	-127.601698232134	-53.8002946056032	396.6082774336302
CV_ITERATIVE	-13.9376	9.57198	-16.7936	-127.261637166946	-53.7338282602455	395.6791547082838
CV_EPNP	-13.9298	9.67549	-16.7999	-127.268562982388	-53.7220578970727	395.8926791544493

What I have learned

I have learned how to estimate the camera pose and state with known camera intrinsic parameters for single camera with only 4 input points in comparison to the many points input used in the calibration step.

Reference:

[1] http://iplimage.com/blog/p3p-perspective-point-overview/
[2] Gao, Xiao-Shan, et al. "Complete solution classification for the perspective-three-point problem." *IEEE transactions on pattern analysis and machine intelligence* 25.8 (2003): 930-943.
[2] http://nghiaho.com/?page_id=671