Real-Time 6DOF Pose Estimation Using Custom Markers

Kartik Paigwar (BT15CSE038), Allala Sreechandra Reddy (BT15CSE009)
Visvesvaraya National Institute of Technology, Nagpur

Abstract

6DOF pose estimation is the problem of determining the position (X, Y, Z) and orientation (Yaw, Pitch and Roll) of the camera relative to the object (or vice-versa). It has been quite researched and studied especially in the area of robotics, for example, Robotic manipulators need to know the exact position and orientation of the object it is supposed to handle. In an autonomous drone delivery system, where drones are expected to land in the human populated areas, this is one of the main tasks to solve, because the delivery locations are not always a fixed position. This work intends to present a general solution to this problem where landing markers must be located and positioned for safe and accurate landing. We designed an L-shaped white colour landing markers for calculating the height and offset distance of the drone relative to this marker. We used the SolvePnPRansac method to compute the pose of the marker using correspondences between 2D image pixels (and thus camera rays) and 3D object points (from the world)

Ⅰ. Perspective-n-Point

Perspective-n-Point is the problem of estimating the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image. The camera pose consists of 6 degrees-of-freedom (DOF) which are made up of the rotation (roll, pitch, and yaw) and 3D translation of the camera with respect to the world. This problem originates from camera calibration and has many applications in computer vision and other areas, including 3D pose estimation, robotics and augmented reality. A commonly used solution to the problem exists for n = 3 called P3P, and many solutions are available for the general case of $n \geq 3$.

a. Problem Specification

Given a set of n 3D points in a world reference frame and their corresponding 2D image projections as well as the calibrated intrinsic camera parameters, determine the 6 DOF pose of

the camera in the form of its rotation and translation with respect to the world. This follows the perspective projection model for cameras:

$s\ P_c = K\ [R \mid T]\ P_w$

Where $P_w = [x\ y\ z\ 1]^T$ is the homogeneous world point, $P_c = [u\ v\ 1]^T$ is the corresponding homogeneous image point, K is the matrix of intrinsic camera parameters, (where fx and fy are the scaled focal lengths, γ is the skew parameter which is sometimes assumed to be 0, and (u_0,v_0) is the principal point), s is a scale factor for the image point, and R and T are the desired 3D rotation and 3D translation of the camera (extrinsic parameters) that are being calculated. This leads to the following equation for the model:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

b. Methods

P3P

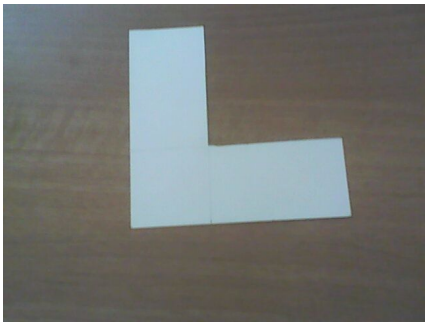
When $n = 3$, the PnP problem is in its minimal form of P3P and can be solved with three-point correspondences. However, with just three point correspondences, P3P yields many solutions, so a fourth correspondence is used in practice to remove ambiguity.

Using RANSAC

PnP is prone to errors if there are outliers in the set of point correspondences. Thus, RANSAC can be used in conjunction with existing solutions to make the final solution for the camera pose more robust to outliers. An open source implementation of PnP methods with RANSAC can be found in OpenCV's Camera Calibration and 3D Reconstruction module in the solvePnPRansac function.

Ⅱ. Markers Specification

Colour : White
Width : 4.5cm
Length : 12 x 12 cm
GrayScale Value : 230



Ⅱ. ROS Camera Calibration

Camera calibration is the process of estimating intrinsic and/or extrinsic parameters. Intrinsic parameters deal with the camera's internal characteristics, such as its focal length, skew, distortion, and image centre. Extrinsic parameters describe its position and orientation in the world. Knowing intrinsic parameters is an essential first step for 3D computer vision, as it allows you to estimate the scene's structure in Euclidean space and removes lens distortion, which degrades accuracy. ROS camera calibration package provides fully automated calibration for checkerboard planar target (see picture below) that can be easily printed on standard sized paper.

CheckerBoard Specification:

- Squares: 8 X 7
- Square Size: 8cm

Camera matrix:

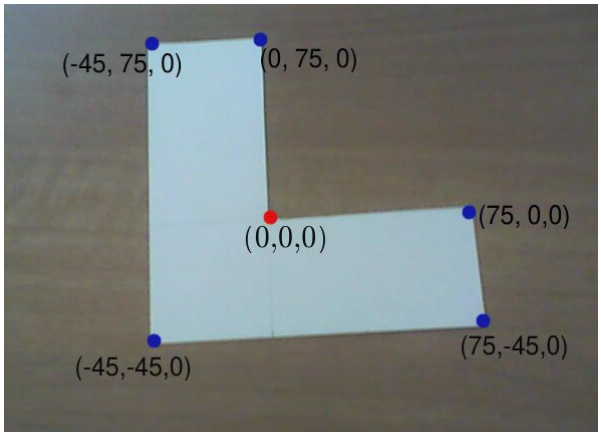
[[707.975174, 0.000000, 326.343126]
[0.000000, 711.961538, 232.354595]
[0.000000, 0.000000, 1.000000]]

Distortion matrix

[0.085624,-0.579964,-0.007464, 0.008612, 0.000000]

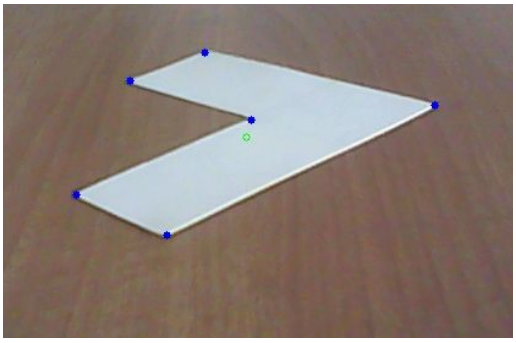
Ⅲ. Calculating 3D World Coordinates

As our marker is of known dimension it is easier to calculate the 3D world coordinates of the corners with respect to the origin considered at the centre innermost corner of the marker.



Ⅳ. Calculating corresponding image coordinates

We thresholded the image to segment the marker. A grayscale threshold value of 230 is being chosen as the marker is white in colour. We then performed morphological Closing operation to remove noisy pixels and fill small holes in the segmented image. Afterwards, we found the contours in the segmented mask and the contour with the largest area is chosen. Then a Polygon is fitted on this contour till 6 corner points are obtained (using the approxPolyDP method of OpenCV).

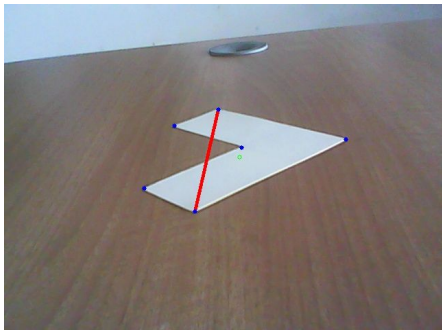


Ⅴ. Finding the inner centre corner

To find the inner corner , we need to find the centroid of the marker first.. We can find the centroid of the object by finding moment of the object in OpenCV. After finding the centroid , we check the distances of all points from this centroid and choose the point which has the least distance as the origin of our coordinate system.

VI . Finding the two farthest corners

To find these corners, We try to join each pair of corners (except the inner corner,which has already been found) with a line and check which line divides the marker such that there are two corner points on either side of the line (including the inner corner). This is the line joining the two farthest points in the marker.



VII. Finding the Outermost corner

After finding the two farthest corners, the outermost corner can be found by checking which side of the line joining the farthest corners contains the origin and then checking which point lies on the same side as this. The point other than the origin is the outermost corner.

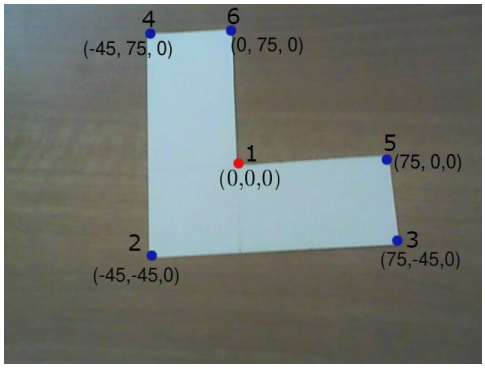
VIII. Finding the other two corners

The other two corners are the points which are on the other side of the line (the side where the origin is not present.).

IX. Ordering the corners

Ordering the corners is necessary since the user needs to know the order in which to measure the world coordinates. The ordering of the corners is as follows :

- Origin(Innermost corner) is labelled as the first corner.
- The outermost corner is labelled as the second corner.
- If the outermost corner is above the origin in the image, then the corner that is towards the right of the outermost corner is labelled as the third corner and the corner that is towards the left of the outermost corner is labelled as the fourth corner. Else, we label the corner towards the left as the third corner and the corner towards the right as the fourth corner.
- The corner nearest to the third corner is labelled as the fifth corner and the remaining corner is labelled as the sixth corner.



X . Getting the pose of the camera

After all the ordered points are obtained in the image, we measure the world co-ordinates in the same order and input them to the solvePnPRansac() algortihm , which solves the equations and outputs the translation and rotation matrices for the camera with respect to the current World reference frame.