
A Practical Guide to Direct Least-Squares (DLS) PnP

*Joel A. Hesch and Stergios I. Roumeliotis
Dept. of Computer Science & Engineering
University of Minnesota*

MARS LAB

Multiple Autonomous
Robotic Systems Laboratory

Technical Report
Number 2011-001, Rev. 1116
Aug. 2011

Dept. of Computer Science & Engineering
University of Minnesota
4-192 EE/CS Building
200 Union St. S.E.
Minneapolis, MN 55455
Tel: (612) 625-2217
Fax: (612) 625-0572
URL: <http://www.cs.umn.edu/~joel>

A Practical Guide to Direct Least-Squares (DLS) PnP

Joel A. Hesch and Stergios I. Roumeliotis
Dept. of Computer Science & Engineering
University of Minnesota

Multiple Autonomous Robotic Systems Laboratory, TR-2011-001, Rev. 1116

Aug. 2011

Abstract

The n -point perspective pose estimation problem (PnP) is a well studied topic in computer vision. PnP seeks to estimate the camera position and orientation (pose) from $n \geq 3$ known points in the scene. In this technical report, we present a practical guide for directly computing all solutions of PnP , by optimally solving a non-linear least-squares cost function for the camera pose.

1 Introduction

The task of solving a nonlinear parameter estimation problem arises frequently in robotics and computer vision. However, despite the plethora of potential applications that exist, the tools we have at our disposal for solving nonlinear estimation problems remain limited. Nonlinear least-squares is one of the most general frameworks we can employ, and under certain noise assumptions, it may even obtain the maximum-likelihood estimate, however, it has several drawbacks. Namely, it typically requires iterative minimization of a nonlinear cost function to reach a solution. Hence, it can easily return a solution which is only locally optimal. Reaching the globally optimal solution is only possible if we have a good initial guess of the solution already.

In this technical report, we examine a relatively new approach to solving nonlinear least-squares by using tools from Algebraic Geometry. In particular, we focus on the problem of n -point pose (position and orientation) determination (PnP) of a camera based on observations of known points in a scene. The key here is that we will transform the nonlinear least-squares cost function into a polynomial. Hence, rendering the optimality conditions as a system of polynomials that can be directly solved using generic polynomial solving tools. Thereby obtaining all of the solutions to our system directly, without the need for iterations or an initial guess.

2 Algorithm Overview

This section provides an algorithm overview of Direct Least-squares (DLS) PnP , which was presented at the International Conference on Computer Vision (ICCV) in 2011 [1]. A block-diagram representation of the method is depicted in Alg. 1.

The key idea is to take the nonlinear least-squares form of PnP [see [1], eq. (3)], and modify it to obtain a cost function which is a polynomial in only the parameters of orientation. Subsequently, we compute the derivatives of the cost function to obtain a polynomial system of equations whose zeros comprise the critical points of the original cost function. The next step is to obtain the Multiplication matrix for that polynomial system, by first forming the Macaulay matrix. Based on the theory of Algebraic Geometry (see [2, 3]), we can obtain all the solutions to the polynomial system via eigen decomposition of the Multiplication matrix.

Using this process, we directly obtain all the optimum points of the nonlinear least-squares cost function directly, without the need for iterations or an initial guess. The approach is efficient with respect to the number of measurements, since both the order and number of variables in the polynomial system that we solve is constant, irrespective of the measurements. Furthermore, since the system structure remains the same across instantiations, we need to only determine the structure of the Macaulay matrix once symbolically.

Algorithm 1 Directly Compute all PnP solutions

Input: Set of n known points observed in 1 image**Output:** All camera pose solutions $\{C_i \mathbf{R}, C_i \mathbf{p}_G\}$ Using the measured points \mathbf{z}_i , construct \mathbf{U} and \mathbf{V} [see [1], eq. (25)]Form all n constraints from [1], eq. (15)Form cost function J' by summing the n constraints**for each** $s_i, i = 1, 2, 3$ **do** Compute $F_i = \nabla_{s_i} J'$, which is the i -th polynomial in the system [1], eq. (17)**end for**Augment the polynomial system with a u -polynomial $F_0 = u_0 + u_1 s_1 + u_2 s_2 + u_3 s_3$ Define the partitioned set of monomials $S = \{S_3, S_2, S_1, S_0\}$ [see [1], eq. (18)]**for each** $S_i, i = 0, \dots, 3$ **do** Extend F_i by multiplying with j monomials in S_i to obtain polynomials $G_{i,j}$ **end for**Using the coefficients of $G_{i,j}, i = 0, \dots, 3, j = 1, \dots, |S_i|$, build the Macaulay matrix \mathbf{M} [see [1], eq. 20]Obtain multiplication matrix $\mathbf{M}_{F_0} = \mathbf{M}_{00} - \mathbf{M}_{01} \mathbf{M}_{11}^{-1} \mathbf{M}_{10}$, via Schur complimentCompute the eigen decomposition of \mathbf{M}_{F_0} to obtain 27 eigen value-vector pairs $(\lambda_i, \mathbf{v}_i)$ **for each** $\mathbf{v}_i, i = 1, \dots, 27$ **do** Normalize \mathbf{v}_i by its first element, v_1 Obtain the i -th solution $[s_1 \ s_2 \ s_3] = [v_2 \ v_4 \ v_{10}]$ Check if solution i is real and a minimum of the cost function using the 2nd derivative test $\nabla_s^2 J' \geq 0$ **end for**Compute position and orientation for each minimum $\{C_i \mathbf{R}, C_i \mathbf{p}_G\}$ using [1] eq. (8) and (10)

3 Results

We evaluated our method experimentally with observations of 7 known points at the corners of a cube. We computed the camera pose with each method (using 3, 4, and 7 known points), and compared the resulting pose value to the MLE estimate obtained using all 7 points. Table 1 lists the errors for orientation and position for each method. We compared our method to competing PnP approaches. Specifically, we considered

- **NPL:** The N-Point Linear (NPL) method of Ansar and Daniilidis [4].
- **EPnP:** The approach of Lepetit *et al.* [5].
- **SDP:** The Semi Definite Program (SDP) approach of Schweighofer and Pinz [6].
- **DLS:** The Direct Least-Squares (DLS) solution presented in this paper. An open source implementation of DLS is available at www.umn.edu/~joel
- **DLS-LM:** Maximum-likelihood estimate, computed using iterative Levenberg-Marquardt (LM) minimization of the sum of the squared reprojection errors, initialized with DLS.

Figure 1 depicts the visual results of the experiment. We show the back-projection of the known global points on the image as green circles, for DLS3 [Fig. 1(a)], and DLS7 [Fig. 1(b)]. In order to further validate the results visually, we also back-project a virtual box (of identical dimensions as the real box) next to the real box. Additional trials are included in the supplemental material [7].

3.1 Processing time comparison

The speed of the four direct methods was evaluated in Matlab 7.8 running on a Linux (kernel 2.6.32) computer with a 2.4 GHz Intel Core 2 Duo processor. NPL and EPnP were the fastest algorithms, requiring approximately 10 ms and 5 ms, respectively, to solve a four-point problem. Our algorithm required approximately 15 ms to compute all local minima of the LS cost function using the Macaulay resultants method. The slowest approach was SDP which required approximately 200 ms to solve the semi-definite program (using SeDuMi). Since the implementations we have for each

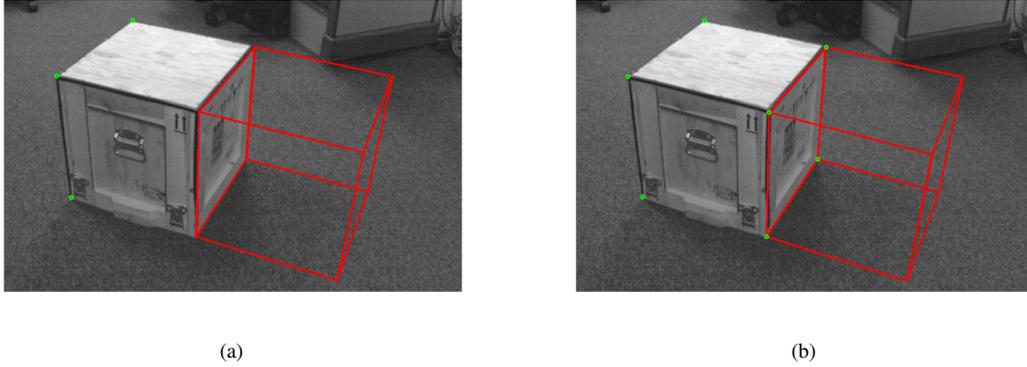


Figure 1: The solution computed using DLS with 3 known points is depicted in 1(a), where the green circles represent the three known points back-projected onto the image using the computed transformation. 1(b) is the result obtained using DLS with 7 known points. In both cases, we also back-project a virtual cube, placed next to the real one, to aid visual verification of the result.

n -points	Ori. Error Norm (rad)	Pos. Error Norm (m)
NPL4	2.87×10^{-3}	8.67×10^{-3}
NPL7	2.12×10^{-3}	2.42×10^{-3}
EP n P4	2.49×10^{-2}	2.33×10^{-2}
EP n P7	1.24×10^{-2}	3.41×10^{-3}
SDP4	4.26×10^{-3}	9.82×10^{-3}
SDP7	3.86×10^{-4}	3.49×10^{-4}
DLS3	5.41×10^{-3}	1.02×10^{-2}
DLS4	4.28×10^{-3}	9.83×10^{-3}
DLS7	4.29×10^{-4}	3.35×10^{-4}

Table 1: The orientation and position errors for different numbers of points. Errors are computed with respect to the MLE estimate of the camera pose computed using all 7 points.

method are Matlab-based and not optimized for speed, we provide these only as “ball-park” figures for performance. Part of our ongoing work is to compare the run-time of these methods using efficient C/C++ implementations.

References

- [1] J. A. Hesch and S. I. Roumeliotis, “A direct least-squares solution for PnP,” in *Proc. of the Int. Conf. on Computer Vision*, Barcelona, Spain, Nov. 6–13, 2011, to appear.
- [2] D. Cox, J. Little, and D. O’Shea, *Ideals, varieties, and algorithms*, 3rd ed. New York, NY: Springer-Verlag, 2000.
- [3] D. A. Cox, J. B. Little, and D. O’Shea, *Using Algebraic Geometry*, 2nd ed. New York, NY: Springer-Verlag, 2005.
- [4] A. Ansar and K. Daniilidis, “Linear pose estimation from points or lines,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 578–589, May 2003.
- [5] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An accurate $O(n)$ solution to the PnP problem,” *Int. Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, Jun. 2008.
- [6] G. Schweighofer and A. Pinz, “Globally optimal $O(n)$ solution to the PnP problem for general camera models,” in *Proc. of the British Machine Vision Conf.*, Leeds, United Kingdom, Sep. 1–4, 2008, pp. 1–10.
- [7] J. A. Hesch and S. I. Roumeliotis, “A practical guide to DLS PnP,” University of Minnesota, Dept. of Comp. Sci. & Eng., MARS Lab, Tech. Rep. 2011-001, Aug. 2011. [Online]. Available: <http://www-users.cs.umn.edu/~joel/>