Jessé Alves
João Luiz Machado Junior
Luis Villamarin

06 November 2022

Professor Florent Nageotte

Pose estimation and 3D registration tutorial report

I. Introduction

The following report aims to evaluate pose estimation PnP problem resolutions from a single perspective projection imaging device, contemplating quantitatively several resolution methods such as ***analytic PnP method*** (provided), ***linear method*** (before and after orthogonality constraints are enforced), and ***numerical optimization*** methods of the reprojection error on pose parameters (independent parameters). *i.e.* (***Levenberg-Marquardt*** and ***quasi-Newton*** optimization) to demonstrate the drawbacks and advantages of each one of those through an estimated error comparison between them. To discuss the results obtained from said methods, this report is divided into a number of subsections: an introduction, a section dedicated to the theory behind pose estimation, an outline of the Matlab script employed and, finally, an analysis of the plots and errors obtained.

During the development of this report, the importance of each of the methods will be seen, as well as the analysis of their performance given the initial conditions established by the exercise, where, for example, the clear advantage of an optimization algorithm Levenberg-Marquardt was observed (average error of 0.47 mm) which, together with analytical resolution method (average error of 1.33 mm), has a lower percentage of error than the other methods mentioned.

II. Theoretical principles

Error Measurement

In order to measure the quality of methods applied, their respective estimation errors were computed, comparing each pose with a known ground truth. Three criterias were used, the mean square error (MAE), the mean absolute error (MSE) and the simplest way that is

comparing directly the components of translation and rotation. The first one (MAE) is calculated through the following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|p_r - p_i\right|, \quad (1)$$

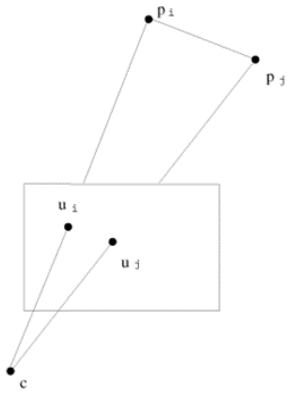
where p_r is the reference point, p_i is the estimate point and n is the number of observations. This criteria eliminates the negative errors, allowing a better analysis of the methods applied. The second criteria (MSE) is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(p_r - p_i)^2, \quad (2)$$

where p_r is the reference point, p_i is the estimate point and n is the number of observations. This method also eliminates negative values for the differences, however, squaring increases the impact of larger errors, in other words, this criteria penalizes larger errors more than smaller errors. Finally, the last method consists in subtracting the translation matrix components and the rotation matrix components.

Perspective n Point problem

This problem consists in using pose estimation principles to figure out the location and orientation of a calibrated camera with respect to the known reference points, starting from a set of markers and trigonometrical relations between 3D reference points and their images [1].



This problem can be solved algebraically, requiring the examination of the relationship between the distances of two points in order to determine the distance between a group of points in the real-world frame by following the equation below.

$$d_{ij}^2 = x_i^2 + x_j^2 - 2x_{ij}cos(\theta_{ij}) \quad (3)$$

This equation allows us to obtain a polynomial system where there are at least 8 solutions related in four pairs. Using Sylvester resultant, it is possible to change the degree of the polynomial expression in order to obtain just a fourth degree equation only dependent on one variable.

$$g(x) = a_5x^4 + a_4x^3 + a_3x^2 + a_2x^1 + a_2x + a_1 = 0 \quad (4)$$

Due to the fact that x_1 is defined by a squared root, and that the values of x_2 and x_3 rely on the value of x_1, this led us to develop a dependent solution, which is why this approach needs at least n = 4 points in order to obtain a unique solution [1].

There are, however, a number of negatives, such as having to actually solve fourth degree polynomials or having to identify the common answer amidst possibly noisy data. The fact that we cannot profit from the data redundancy, which should boost stability, is the final and maybe most significant factor. Once those distances have been recovered, the problem becomes a 3D localization problem that can be solved in several other ways such as with Horn's method [2], the approximation with 3 points, and linear approaches.

Enforcing orthogonality

Studies in robotics [3] and computer vision [4], [5] observe that the rotation matrix:

$$\boldsymbol{R} = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix},$$

is an orthogonal matrix, since each column vector of matrix **R** represents the unit vectors of an orthonormal frame, in other words, the column vectors in matrix **R** are mutually orthogonal [6]. Consequently, the matrix **R** has the following properties:

1. $R^TR = I$, where I denotes the identity matrix;
2. $R^{-1} = R$

3. $det(R) = 1$, if the frame is right-handed, while $det(R) = -1$, if the frame is left-handed.

Based on this, in order to obtain a rotation matrix in the pose estimation process that respects the internal constraints, it is sometimes necessary to verify and enforce the orthogonality. In this tutorial, the method used consists in finding the nearest orthonormal matrix R of the nonsingular matrix M found in pose estimation processes. It can be demonstrated that the matrix R, subject to the condition $R^TR = I$, that minimizes

$$\sum_{i=1}^{3}\sum_{j=1}^{3}\left(m_{i,j} - r_{i,j}\right)^2 = Tr[(M-R)^T(M-R)] =$$

$$= Tr(M^TM) - 2Tr(R^TM) + Tr(R^TR),$$

and therefore, maximizes the trace

$$Tr(R^TM),$$

occurs in the decomposition of M into the product of an orthonormal matrix and a symmetric matrix [2]. Then, the rotation matrix $R = M(M^TM)^{-1/2}$, with $M = USV^T$ obtained:

- If $det(UV^T) > 0$ implies that $R = UV^T$
- And if $det(UV^T) < 0$ implies that

$$R = U\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & -1\end{bmatrix}V^T.$$

Numerical optimization

For the numerical optimization method of pose estimation, two minimization functions from Matlab were selected: *fminunc* and *lsqnonlin*. The first is defined in the documentation as a nonlinear programming solver utilized to find the minimum of a given function $f_{(x)}$, starting at a provided initial value for x. In the circumstances of this tutorial, *fminunc* is used to run a number of iterations until it finds a local minimum for the difference between the reference points in the camera frame (m_i*) and the reprojected points (m_i) in the form:

$$\sum_{i=1}^{n}\left|\left|m_i - m_i^*\right|\right|^2. \quad (5)$$

Similarly, *lsqnonlin* is defined in the documentation as a solver for nonlinear least-squares curve fitting problems. Unlike *fminunc*, it only requires describing the function to be minimized as the difference between m_i and m_i*, returning a vector of values. The sum of squares of said values is computed implicitly in the algorithm.

III. Script outline - Methodology

In order to evaluate the aforementioned solution strategies for pose estimation problems, this tutorial features a set of Matlab functions, data files and scripts, all centered around a set of points in R^3 space and their corresponding projections into images in pixels. Data which is necessary for the computations, such as the coordinates of the object points, the intrinsic parameters matrix K and a number of different test configurations for the camera, has been provided in a simple, immediately accessible data file. Similarly, one of the pose estimation solutions, the Efficient PnP, has already been provided and does not need to be completed. The main task of computing and comparing solutions to the PnP problem is, in the code, divided into four main sections.

1. **The reference data for comparisons**, which is concerned with applying the provided data to generate both the perfect image points *and* image points modified through the addition of noise. Five noise amplitudes were specified (0, 0.1, 0.5, 1 and 2), therefore generating five sets of noise-shifted image points for use in the comparative analysis of pose estimation solutions.

2. **The error computations for each PnP solution**, which provide both numerical data (median and maxi errors) and plots for better visualization of the translation and image reprojection errors for all the noise amplitudes. This section also includes and already evaluates the performance of the ePnP method which, as mentioned before, does not need to be modified.

3. **The linear solution to the PnP problem**, both with and without enforcing orthogonality constraints. This section employs and then evaluates the results of a function for a linear solving method. Given the points both in the object (P^{obj}) and camera frame (m_i), it is possible to write the effect of the homogeneous transformation as:

$$\left[x_k\ y_k\right]^T = \left[\frac{r_1 P_i^{obj} + t_x}{r_3 P_i^{obj} + t_z}\ \frac{r_2 P_i^{obj} + t_y}{r_3 P_i^{obj} + t_z}\right]^T, \tag{6}$$

Where x_k and y_k represent the coordinates of each point in the camera frame, r_1, r_2 and r_3 represent the three lines of the rotation matrix and t_x, t_y and t_z represent the translation associated with the pose. With this arrangement, the problem is thus easily composed into a system of linear equations:

$$\begin{aligned} \left(r_3 P_i^{obj} + t_z\right)x_k - \left(r_1 P_i^{obj} + t_x\right) &= 0 \\ \left(r_3 P_i^{obj} + t_z\right)y_k - \left(r_2 P_i^{obj} + t_y\right) &= 0 \end{aligned} \tag{7}$$

An estimate for the pose can then be obtained by solving the system as a matrix equation $Ax = 0$, for instance by employing singular value decomposition (SVD) on A and taking x as the right-singular vector corresponding to the smallest singular value. Such a result still needs to be scaled, however, in order to ensure that the rotation has a unitary determinant. The final step for this section is to also provide an estimate for the pose with a rotation matrix that respects the orthogonality constraints of a rotation matrix, which can be easily done using the method discussed in the previous section.

4. **The numerical optimization solution**, which re-writes the pose estimation problem as a least-squares minimization problem, i.e., the problem of minimizing the error between the reference points and the points reprojected from the initial estimate of the pose. The starting configuration was established with Euler angles, to preserve orthogonality, and a translation was chosen as equal to the centroid of the points in the object frame. With it, this section then employs either one of the two possible minimization functions (*lsqnonlin* and *fminunc*) to run through a number of iterations until a satisfactory approximation is reached. For this exercise in particular, the minimization algorithm for *lsqnonlin* was pre-defined as the Levenberg-Marquardt algorithm.

IV. Results and analyses

Having described the nature of the problem, as well as the general structure of the program executed in Matlab, we may now proceed to analyze the results obtained by evaluating each of the four highlighted methods with a number of different metrics, a feat done primarily by interpreting plots and numerical measurements obtained from the program. One important point to emphasize is the fact that, for the plots showing errors per noise level, the linear method prior to enforcing orthogonality constraints was labeled “linear wo norm”, contrary to the original labeling in the script skeleton.

One of the metrics acquired to compare the methods was the execution time, measured with the start (tic) and stop (toc) stopwatch timer functions in Matlab. The results were compiled and are presented in *Table 1*.

Method	0 pix noise	0.1 pix noise	0.5 pix noise	1 pix noise	2 pix noise	μ
Linear	0.1440 s	0.0968 s	0.1279 s	0.1269 s	0.1032 s	0.1198 s
Analytic	0.7329 s	0.3941 s	0.5697 s	0.3342 s	0.8015 s	0.5665 s
Optimization	4.0250 s	2.0963 s	2.2783 s	2.2635 s	1.9313 s	2.5189 s

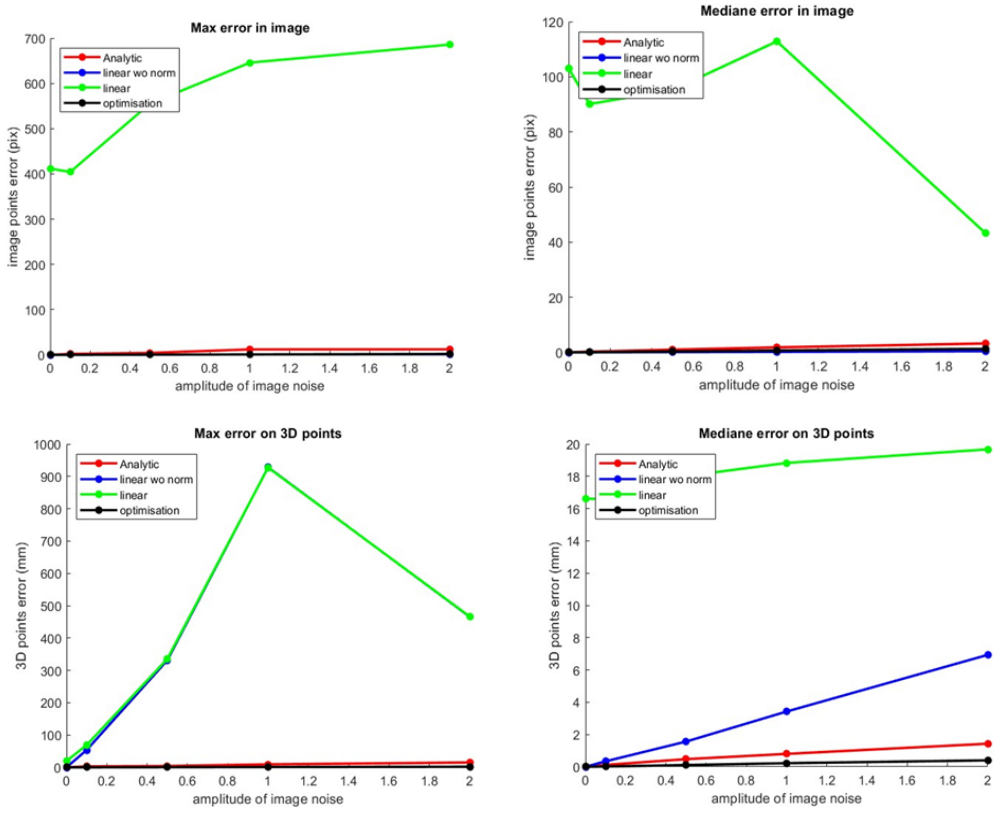
Table 1. Duration (in seconds) of each pose estimation method.

Table 1 presents the computation time required by the linear, analytic and numerical optimization methods to compute estimates for all 271 different test configurations at each noise level. The analytic solution averages a time of 0.5665 seconds per set of configurations, or approximately 0.002 seconds for each individual pose. Meanwhile, the linear method stands out as the fastest (0.1198 seconds on average, 0.0004 seconds per pose), while the numerical optimization method predictably takes the last place as the slowest (2.5189 seconds on average, 0.01 seconds per pose).

1) Error comparison between the four methods

As mentioned in the methodology, with the objective of checking the response quality of different problem-solving algorithms for the pose estimation of an object, five noise magnitudes were added (0, 0.1, 0.5, 1 and 2 pixels respectively). To better address the performance of each algorithm, the errors have been separated between: the errors of the images produced, the 3D points and finally the translations.

Due to the random nature of noises, for every execution all the maximum error results will vary, nevertheless, Mediane error will remain with a consistent magnitude. Hence, in this analysis, the results of the plots on the left (maximums) are just for illustrative purposes and a general behavior will be described below.



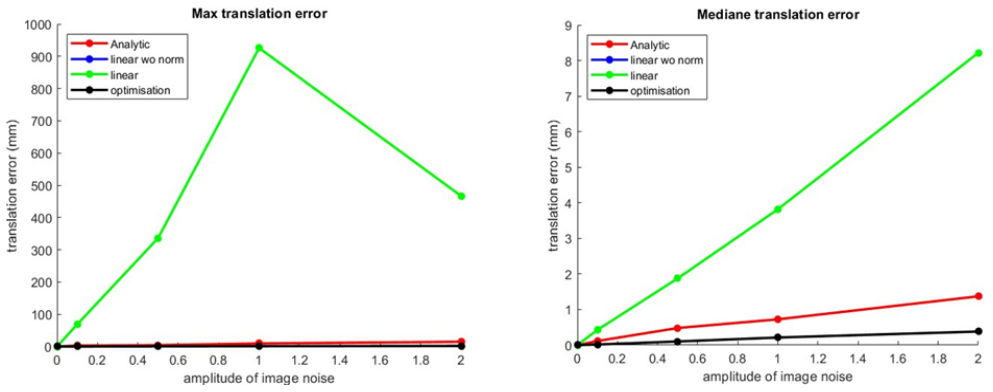


Fig 1. Comparison of the maximum (left) and average (right) errors of the images of the points (in pixels) (first line) , errors positions of the 3D points in the object frame (mm) (second line) and finally, errors positions of the translation (mm) (third line) according to the amplitude of the noise added for each one of the pose estimation algorithms used in this report (Analytic PnP, Linear before and after enforcing orthogonality and numerical optimisation Levenberg-Marquardt).

In this segment, we see that the highest maximum error (left) for *Fig 1.* is given by the Linear algorithms before and after enforcing orthogonality, having a significant increase in its growth rate slope when the noise is between 0.1 and 1, being a clear indicator of the sensitivity of this method to external noise.

However, this is not the case for the mean of the errors in the previous scenarios, where (despite there are some other more efficient approaches as will be commented further), there is a clear difference in the results when an orthogonality enforcing has been previously performed for the linear resolution method. This, because the error for the 3D points maintains a difference between 16 and 18 mm (for the ones that have orthogonality enforcement) than a calculation that does not use SVD, which has a range between 0 and 6 mm, which is a reason why this method should be used with caution due to as a consequence of some precision errors: it can be dangerous to be used in a real environment.

We may, in conclusion, highlight a few limitations particular to the simple linear resolution method. Firstly, given that the problem presents 12 unknowns, a minimum of 6 points are required, not accounting for collinear points. Most importantly, the method also exhibits a certain degree of instability, demonstrating a limitation when compared with the numerical methods, for example. In fact, we observe that, for the linear method employed, the difference in errors before and after enforcing orthogonality constraints comes from the application of Singular Value Decomposition, as discussed in section II. After all, SVD provides the nearest

approximation of a given matrix, ensuring a rotation that has physical meaning, but enforcing orthogonality to satisfy these internal constraints implies in reducing the accuracy in the presence of noise, creating a noticeable disparity.

On the other hand, it is possible to see that for the provided analytical PnP, together with the numerical optimization method with Levenberg-Marquardt algorithm, have the best results when comparing with the calculated poses and the positions of the objects in real space, with almost null errors for each one of the plots in *Fig 1*, giving indications of robustness due to their low sensitivity to noise (contrary to the linear method).

In all the figures, the algorithm that stands out against all, having a total of errors very close to zero for each of the generated random noise amplitudes, is the Levenberg-Marquardt algorithm. It is as such because it is able to satisfactorily combine two numerical minimization algorithms, such as Gauss-Newton method, which assumes that the least squares function is locally quadratic, and additionally uses the gradient descent as a way to update the parameters in the steepest-descent direction [7].

2) Group error graphs by noise amplitude

As mentioned in the last section, three types of errors were computed. In order to analyze these different methods, the graphics below were generated.

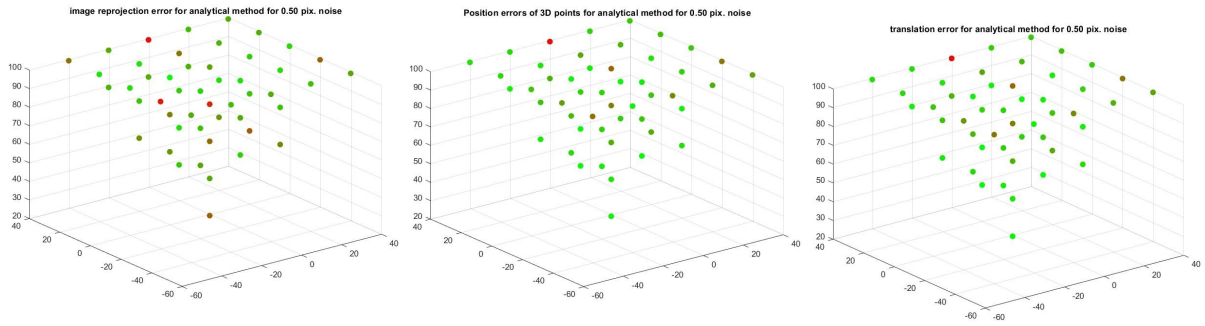


Fig 2. (From Right to Left) Errors of the produced images , errors of the 3D points and Errors of the translations.

The logic behind these graphics lies in measuring the error magnitude through a color scale. Where the red color registers the high errors and the green color registers the low errors. It is possible to verify that the image reprojection error shows higher errors than the other two graphics, exhibiting more dots near the red color. As the level of noise and the considered points

are the same, the reason for it is that the image reprojection error is computed in pixels, making this method more sensitive to low variations, when compared to methods that are computed in millimeters. Therefore, the pixel approach shows an accuracy analysis for small errors.

3) Error per axis for each one of the test configurations in the analytical method

Besides the error plots and spatial distributions, this set of programs also generates plots showcasing the different translation errors along individual axes for one method in particular (the ePnP) with respect to each of the five possible noise amplitudes.

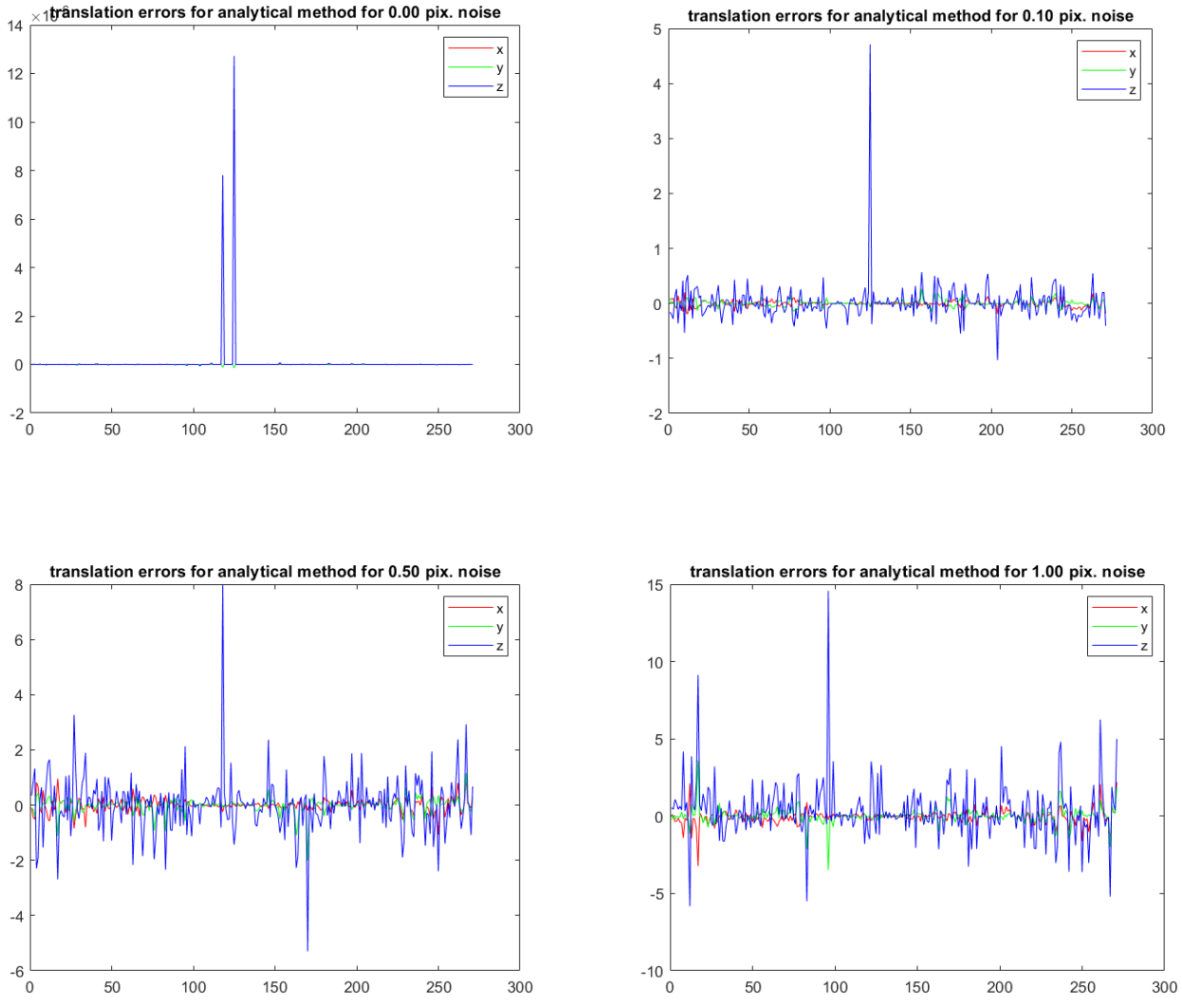


Fig 3. Translation errors along axes x, y and z for the analytical method at different noise amplitudes

From what has already been discussed in the previous section, the provided analytical solution has consistently good results for estimating poses, although it is still affected by both unusual configurations and noisy data. For smaller noise levels, there is a visible spike in the translation error in the z axis for test configurations such as 111, 118, 125 and 153, all of which differ from their immediate neighbors by featuring no rotation at all.

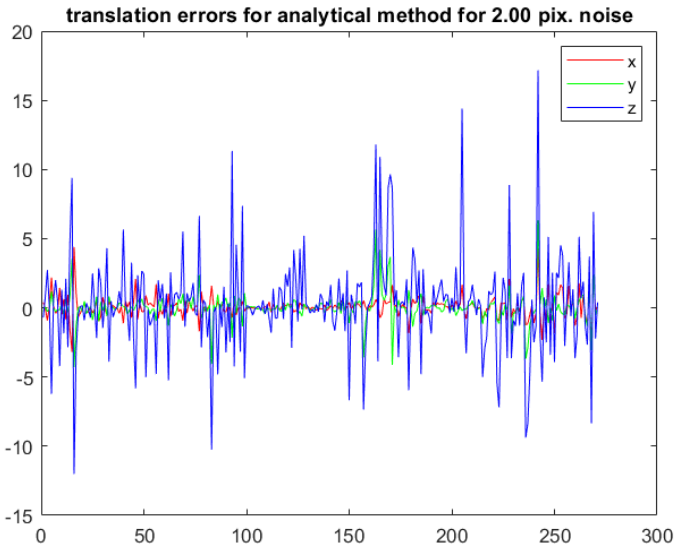


Fig 4. Translation errors along axes x, y and z for the analytical method at the highest noise amplitude

For higher noise levels, spikes in the error appear mostly for test configurations with unitary antidiagonals in the rotation matrix, such as in configurations 77, 93, 99, 236 and 260. Nonetheless, for all cases, said errors sit well within an acceptable range, particularly when considering the data discussed in the previous stages of this report and the errors for the other methods.

Finally, it has been considered pertinent to summarize the performance of the proposed methods in a qualitative way in order to better understand their purpose and functionality for possible applications. Said summary is shown in *Table 2*.

It is important to take into account three aspects: the speed of execution of the algorithm in order to make quick decisions if required, its stability against external conditions such as noise, and finally its reliability with respect to the ability to rebuild objects with as little error as possible.

Table 2. Qualitative summary of the performance of each of the algorithms presented during the report

V. Conclusion

As has been observed through the different results, despite their limitations, all the methods are useful under certain conditions for determining the pose estimation problem, yielding error results that may vary in magnitude.

For example, due to the sensitivity to noise, the *linear method*, despite its speed of execution (approx.5 times faster than Analytic PnP and 25 times faster than LM Optimisation), is not the first option to solve a real problem because the errors in distance are in average ±10 mm with respect to the original object, being the most imprecise of all when having non ideal scenarios.

Lower errors were obtained by the Analytic PnP and the Optimisation LM method, being this last one who, despite taking longer to execute (approx. 5 times more than Analytic PnP), has very low errors (got twice less the error than Analytic PnP) as shown in Fig 1.

The foregoing leads us to think that depending on the requirements and needs of each particular case, it is necessary to review the performance aspects evaluated (such as speed of execution and precision) in this report in order to make a decision when choosing an algorithm.

References

[1] L. Quan and Z. Lan. Linear n-point camera pose determination. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(8), 1999.

[2] B. Horn, H. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. Journal of the Optical Society of America, 5(7) :1127–1135, 1987.

[3] S. Cubero. Industrial Robotics Theory, Modelling and Control. Pro literatur Verlag Robert Mayer-Scholz, ISBN: 3-86611-285-8, 2007.

[4] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN : 0521540518, second edition, 2004.

[5] R. Horaud and O. Monga. Vision par ordinateur. Hermes, 1995.

[6] L. Sciavicco and B. Siciliano, Modeling and Control of Robot Manipulators, Springer, 1996.

[7] H. Gavin. The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems, 2020.