Optimizing the Solution of PNP

December 6, 2022

1 Introduction

The PNP problem is very important and well-known, and many papers appeared which discuss it. The problem is to determine the rotation and translation of a camera, when given correspondences between points in the "real world" and their location in the image captured by the camera.

The solution which is considered the most accurate is the one presented in many papers, for example conf/bmvc/SchweighoferP08 (which was followed by many papers). In this solution method, the error function (which measures how far the real-world points are from the line of projection passing through the camera pinhole and the image points), squares this error, and adds them for all the points. This error depends on both the translation and rotation. Then, the derivative by the translation term is computed, and the solution which makes the derivative equal to 0 computed, and back-substituted into the original error expression. This yields an expression to minimize which depends only on the rotation, and can be expressed as a second-order polynomial in the elements of the rotation matrix R, when it is "flattened" and viewed as a vector of length 9:

$$RMR^T \tag{1}$$

where M is a semidefinite positive matrix, which depends on the input (3D and 2D points).

To minimize Eq. 1, we can proceed by solving an optimization problem with the 9 elements of R, subject to 6 constraints which force it to be a rotation matrix (the rows define an orthogonal matrix, so all their norms are equal to 1 and the inner product of every two is zero). But a better

approach (with only one constraint), which was used in many papers, is to represent R as second-degree polynomials of only four parameters (a unit "quaternion" elements), and then to minimize over these elements, with only one constraint: the sum of their squares is equal to 1. This yield an error as in Eq. 1, but now the 9 elements of R are second-degree polynomials in four variables, x, y, z, w, and Eq. 1 should be minimized with the constraints $x^2 + y^2 + z^2 + w^2 = 1$. Mathematically, this is equivalent to minimizing a fourth-degree polynomial, $p_4(x, y, z, w)$, on the "three dimensional unit sphere", denoted by $S^3 = \{(x, y, z, w) \in \mathcal{R}^4 | x^2 + y^2 + z^2 + w^2 = 1\}$.

Even in this simplified form, there is no known solution which is fast, especially for real-time drone navigation and control. Directly solving the polynomial on software packages like Maple and Gloptipoly requires a very long time, which we need a solution that runs in miliseconds! Therefore, various heuristics are used. The one which is considered the best (in terms of accuracy) is the following, which is based on the famous "Lasserre Hierarchy" for minimizing polynomials:

$$\max \gamma \text{ such that } p_4(x, y, z, w) - \gamma = (1 - x^2 - y^2 - z^2 - w^2) q_2(x, y, z, w) + (1, x, y, z, w, x^2 \dots w^2) B (1, x, y, z, w, x^2 \dots w^2)^T$$

for $(1, x, y, z, w, x^2 \dots w^2) = (1, x, y, z, w, x^2, xy, xz, xw, y^2, yz, yw, z^2, zw, w^2)$ and B a semidefinite positive 15×15 matrix.

The intuition is: suppose that this equality holds. Then, on S^3, $1 - x^2 - y^2 - z^2 - w^2 = 0$, so on S^3, $p_4(x, y, z, w) - \gamma$ is equal to vBv^T, and as well known, this is always positive if B is a semidefinite positive matrix. So, on S^3, γ is a lower bound for $p_4(x, y, z, w)$, and we search for the maximal lower bound, which is equal to the minimum.

It must be stated that this approach does not always yield the correct minimum, for example for the famous "Choi-Lam polynomial", which is equal to $w^4 + x^2y^2 + x^2z^2 + y^2z^2 - 4xyzw$, but *cannot* be written as

$$(1-x^2-y^2-z^2-w^2) q_2(x, y, z, w) + (1, x, y, z, w, x^2 \dots w^2) B (1, x, y, z, w, x^2 \dots w^2)^T$$

Even for the PNP problem, there are cases in which it fails, as proved in journals/sensors/AlfassiKR21. However these cases seem to be very rare, as also stated in journals/jmiv/BrynteLIOK22. In the second part of this report we describe a method which always finds the solution, with a guarantee on the approximation; this method is slower than what we describe next.

Formulating as in Eq. 2 leads to a *semidefinite programming* (SDP), in which γ must be maximized, under the following constraints:

1. B is a semidefinite positive matrix.
2. 70 equalities must be satisfied, which correspond to the 70 coefficients of $p_4(x, y, z, w)$.
3. The above equalities also depend on the 15 coefficients of the second-degree polynomial $q_2(x, y, z, w)$.

2 Proposed Improvement

The problem described in the introduction can be solved using available packages (SDPA, HSDP and others), or also by direct optimization (as in journals/corr/abs-2203-02686 which was also presented in the IROS 2022 conference). Here we present a solution which is also based on SDP, with an approach that resembles the one described above and followed in many papers, but which is simpler in the the following sense:

1. There are only 34 variables, not 85.
2. There are no equality and no inequality constrains.
3. There are no "auxiliary variables" like $q_2(x, y, z, w)$.
4. The matrix B is 10×10, not 15×15.

Since a major part of the solution is computing the inverse of the Hessian matrix (which is $O(n^3)$ for a matrix of size $n \times n$), and our Hessian is quite smaller, our proposed solution is about 10 times faster than previous work (it takes about 1 millisecond, on a relatively weak i7-6500u processor, to find the minimum).

2.1 Details of the proposed solution

The solution uses the fact that the polynomial $p_4(x, y, z, w)$ is homogeneous, meaning it contains only monomials with total degree 4, and all the rest have coefficients equal to 0.

Suppose that the minimum of $p_4(x, y, z, w)$ on S^3 is γ. Since on S^3 we have $x^2 + y^2 + z^2 + w^2 = 1$, then on S^3

$$q_4(x, y, z, w) = p_4(x, y, z, w) - \gamma(x^2 + y^2 + z^2 + w^2)^2 \geq 0$$

But $q_4(x, y, z, w)$ is also homogeneous (contains only monomials of total degree 4), hence we have, for every real number α:

$$q_4(\alpha x, \alpha y, \alpha z, \alpha w) = \alpha^4 q_4(x, y, z, w)$$

but for every $(x, y, z, w) \in \mathcal{R}^4$, $\frac{(x,y,z,w)}{||(x,y,z,w)||} \in S^3$, and $q_4(x, y, z, w) = ||(x, y, z, w)||^4 q_4(\frac{(x,y,z,w)}{||(x,y,z,w)||})$, therefore $q_4(x, y, z, w) \geq 0$ on S^3 iff $q_4(x, y, z, w) \geq 0$ on $\mathcal{R}^4$.

Next, we make the following assumption, which is similar to the one made when solving PNP: that, since $q_x(x, y, z, w) \geq 0$ in $\mathcal{R}^4$, it can be written as

$$q_4(x, y, z, w) = (x^2, xy \dots w^2) B (x^2, xy \dots w^2)^T \tag{2}$$

for a 10×10 matrix B.

Therefore, we can replace the previous optimization problem by

$$\max \gamma \text{ such that } p_4(x, y, z, w) - \gamma(x^2 + y^2 + z^2 + w^2)^2 = (x^2, xy \dots w^2) B (x^2, xy \dots w^2)^T$$

The size of the problem can be further reduced as follows. By equating the coefficient of x^4, we see that $a_{4000} - \gamma = B_{1,1}$, where a_{4000} is the coefficient of x^4 in $p_4(x, y, z, w)$. Since we wish to maximize γ, we can minimize $B_{1,1}$ (because a_{4000} is fixed). So now the problem becomes

$$\min B_{1,1} \text{ such that } p_4(x, y, z, w) - (a_{4000} - B_{1,1})(x^2 + y^2 + z^2 + w^2)^2 = (x^2, xy \dots w^2) B (x^2, xy \dots w^2)^T$$

This problem has no inequality constraints, and can be described as the problem of minimizing $\mathrm{Tr}(CB)$, where C is the 10×10 matrix with $C_{1,1} = 1$ and all other elements equal to 0, under 34 equality constraints on B, corresponding to the 34 coefficients (we don't need one for $B_{1,1}$). In the SDP notation, we represent these conditions as the trace of the product of B with the corresponding matrices. The matrices are defined by the following, with all other elements equal to 0:

A0004[10, 10] = 1
A0013[9, 10] = 1

A0013[10, 9] = 1
A0022[8, 10] = 1
A0022[9, 9] = 1
A0022[10, 8] = 1
A0031[8, 9] = 1
A0031[9, 8] = 1
A0040[8, 8] = 1
A0103[7, 10] = 1
A0103[10, 7] = 1
A0112[6, 10] = 1
A0112[7, 9] = 1
A0112[9, 7] = 1
A0112[10, 6] = 1
A0121[6, 9] = 1
A0121[7, 8] = 1
A0121[8, 7] = 1
A0121[9, 6] = 1
A0130[6, 8] = 1
A0130[8, 6] = 1
A0202[5, 10] = 1
A0202[7, 7] = 1
A0202[10, 5] = 1
A0211[5, 9] = 1
A0211[6, 7] = 1
A0211[7, 6] = 1
A0211[9, 5] = 1
A0220[5, 8] = 1
A0220[6, 6] = 1
A0220[8, 5] = 1
A0301[5, 7] = 1
A0301[7, 5] = 1
A0310[5, 6] = 1
A0310[6, 5] = 1
A0400[5, 5] = 1
A1003[4, 10] = 1
A1003[10, 4] = 1
A1012[3, 10] = 1
A1012[4, 9] = 1

A1012[9, 4] = 1
A1012[10, 3] = 1
A1021[3, 9] = 1
A1021[4, 8] = 1
A1021[8, 4] = 1
A1021[9, 3] = 1
A1030[3, 8] = 1
A1030[8, 3] = 1
A1102[2, 10] = 1
A1102[4, 7] = 1
A1102[7, 4] = 1
A1102[10, 2] = 1
A1111[2, 9] = 1
A1111[3, 7] = 1
A1111[4, 6] = 1
A1111[6, 4] = 1
A1111[7, 3] = 1
A1111[9, 2] = 1
A1120[2, 8] = 1
A1120[3, 6] = 1
A1120[6, 3] = 1
A1120[8, 2] = 1
A1201[2, 7] = 1
A1201[4, 5] = 1
A1201[5, 4] = 1
A1201[7, 2] = 1
A1210[2, 6] = 1
A1210[3, 5] = 1
A1210[5, 3] = 1
A1210[6, 2] = 1
A1300[2, 5] = 1
A1300[5, 2] = 1
A2002[1, 10] = 1
A2002[4, 4] = 1
A2002[10, 1] = 1
A2011[1, 9] = 1
A2011[3, 4] = 1
A2011[4, 3] = 1

A2011[9, 1] = 1
A2020[1, 8] = 1
A2020[3, 3] = 1
A2020[8, 1] = 1
A2101[1, 7] = 1
A2101[2, 4] = 1
A2101[4, 2] = 1
A2101[7, 1] = 1
A2110[1, 6] = 1
A2110[2, 3] = 1
A2110[3, 2] = 1
A2110[6, 1] = 1
A2200[1, 5] = 1
A2200[2, 2] = 1
A2200[5, 1] = 1
A3001[1, 4] = 1
A3001[4, 1] = 1
A3010[1, 3] = 1
A3010[3, 1] = 1
A3100[1, 2] = 1
A3100[2, 1] = 1
A4000[1, 1] = 1
A0400[1, 1] = -1
A0040[1, 1] = -1
A0004[1, 1] = -1
A2200[1, 1] = -2
A2020[1, 1] = -2
A2002[1, 1] = -2
A0220[1, 1] = -2
A0202[1, 1] = -2
A0022[1, 1] = -2

Next, we switch to the dual problem, with the well-known method of replacing

$$\min \mathrm{Tr}(CB) \text{such that } \mathrm{Tr}(A_i B) = b_i$$

with

$$\max b_1 y_1 + \ldots b_k y_k \text{ such that } C - b_1 A_1 - \ldots b_k A_k \text{ is an SPD matrix}$$

Eventually we obtain the following problem:
Maximize

$$y34\,a3100 + y33\,a3010 + y11\,a0211 + y12\,(a0220 - 2a4000) + y13\,a0301 + y14\,a0310 + y15\,(a0400 - a4000) + y16\,a1003 + y17\,a1012 + y18\,a1021 + y19\,a1030 + y32\,a3001 + y31\,(a2200 - 2a4000) + y30\,a2110 + y29\,a2101 + y20\,a1102 + y28\,(a2020 - 2a4000) + y21\,a1111 + y22\,a1120 + y23\,a1201 + y24\,a1210 + y25\,a1300 + y26\,(a2002 - 2a4000) + y27\,a2011 + y10\,(a0202 - 2a4000) + y9\,a0130 + y8\,a0121 + y7\,a0112 + y6\,a0103 + y5\,(a0040 - a4000) + y4\,a0031 + y1\,(a0004 - a4000) + y2\,a0013 + y3\,(a0022 - 2a4000)$$

such that

$$Y = \begin{bmatrix} Y_{11} & -y34 & -y33 & -y32 & -y31 & -y30 & -y29 & -y28 & -y27 & -y26 \\ -y34 & -y31 & -y30 & -y29 & -y25 & -y24 & -y23 & -y22 & -y21 & -y20 \\ -y33 & -y30 & -y28 & -y27 & -y24 & -y22 & -y21 & -y19 & -y18 & -y17 \\ -y32 & -y29 & -y27 & -y26 & -y23 & -y21 & -y20 & -y18 & -y17 & -y16 \\ -y31 & -y25 & -y24 & -y23 & -y15 & -y14 & -y13 & -y12 & -y11 & -y10 \\ -y30 & -y24 & -y22 & -y21 & -y14 & -y12 & -y11 & -y9 & -y8 & -y7 \\ -y29 & -y23 & -y21 & -y20 & -y13 & -y11 & -y10 & -y8 & -y7 & -y6 \\ -y28 & -y22 & -y19 & -y18 & -y12 & -y9 & -y8 & -y5 & -y4 & -y3 \\ -y27 & -y21 & -y18 & -y17 & -y11 & -y8 & -y7 & -y4 & -y3 & -y2 \\ -y26 & -y20 & -y17 & -y16 & -y10 & -y7 & -y6 & -y3 & -y2 & -y1 \end{bmatrix}$$

where $Y_{11} = 2y26 + 2y28 + 1 + 2y31 + y1 + 2y3 + y5 + 2y10 + 2y12 + y15$ is an SDP matrix
This is a much smaller problem than the existing one for solving the SDP problem. We have tested various packages, and the fastest one turned out to be SDPA (version 7.3.16).

2.2 Rank 1 Constraint

In all our experiments, the matrix Y turned out to be of rank 1. This is not surprising, as it is known that the solutions correspond to the moments of the point at which the minimum of $p_4(x, y, z, w)$ is obtained, hence Y also equals

$$\begin{bmatrix}
x^4 & x^3y & x^3z & x^3w & x^2y^2 & x^2yz & x^2yw & x^2z^2 & x^2zw & x^2w^2 \\
x^3y & x^2y^2 & x^2yz & x^2yw & x\,y^3 & x\,y^2z & x\,y^2w & xy\,z^2 & xyzw & xy\,w^2 \\
x^3z & x^2yz & x^2z^2 & x^2zw & x\,y^2z & xy\,z^2 & xyzw & x\,z^3 & x\,z^2w & xz\,w^2 \\
x^3w & x^2yw & x^2zw & x^2w^2 & x\,y^2w & xyzw & xy\,w^2 & x\,z^2w & xz\,w^2 & x\,w^3 \\
x^2y^2 & x\,y^3 & x\,y^2z & x\,y^2w & y^4 & y^3z & y^3w & y^2z^2 & y^2zw & y^2w^2 \\
x^2yz & x\,y^2z & xy\,z^2 & xyzw & y^3z & y^2z^2 & y^2zw & y\,z^3 & y\,z^2w & yz\,w^2 \\
x^2yw & x\,y^2w & xyzw & xy\,w^2 & y^3w & y^2zw & y^2w^2 & y\,z^2w & yz\,w^2 & y\,w^3 \\
x^2z^2 & xy\,z^2 & x\,z^3 & x\,z^2w & y^2z^2 & y\,z^3 & y\,z^2w & z^4 & z^3w & z^2w^2 \\
x^2zw & xyzw & x\,z^2w & xz\,w^2 & y^2zw & y\,z^2w & yz\,w^2 & z^3w & z^2w^2 & z\,w^3 \\
x^2w^2 & xy\,w^2 & xz\,w^2 & x\,w^3 & y^2w^2 & yz\,w^2 & y\,w^3 & z^2w^2 & z\,w^3 & w^4
\end{bmatrix} \tag{3}$$

which is easily seen to be of rank 1. Since every rank 1 matrix can be represented as vv^T, for some column vector v, we could reduce the problem to one with only 10 variables! However, there are no known methods to solve it, as it is not convex, so we abandoned this method.

2.3 Finding a Good Starting Point

It is known that finding a good initial point can substantially reduce the running time of SDP. We tried a few methods, including running a random problem for about half the number of iterations, and taking the result as an initial point. What eventually worked best was to use as a starting point the matrix Y in Eq. 3 which is defined as the average of many random points over S^3. This provides a matrix which is very well conditioned (has a relatively large determinant), and on the average it reduced the running time by a factor of 2.

3 Solution with "Slices"

As explained previously, the method for solving the PNP optimization problem by assuming that a polynomial which obtains only positive values can be written as vBv^T, where v is a vector of monomials and M an SPD matrix, is not guaranteed to work, although its failure cases are very rare. However, due to a famous theorem of Hilbert, this assumption is true for polynomials

with three variables, x, y, z. That means that a homogeneous polynomial of degree 4 in x, y, z can always be written as

$$(x^2, xy, xz, y^2, yz, z^2)B(x^2, xy, xz, y^2, yz, z^2)^T$$

for some 6×6 SDP matrix B. This allows a solution as we presented in Section 2.1, but which is always correct, and runs very fast, as it can be written as an optimization problem in 10 variables and a 6×6 matrix.

In order to reduce the number of variables from 4 to 3, we look at "slices" of the form $w = az$, and their intersections with S^3. Obviously, the more slices we have, the closer the minimum on them is to the real minimum.

To test the accuracy of this method, we used inputs for PNP (supplied by Yuval Alfassi), and ran for different numbers of slices. We tried both $2n$ slices of the form above ($w = az$), and also n slices both of the form $w = az$ and $y = ax$. The first method worked better on the average.

The accuracy rapidly increases with the number of slices; here are the results for the relative error for various numbers of slices (average over many inputs):

40 slices: 0.0014
100 slices: 0.00022
200 slices: 0.000061
400 slices: 0.000017

The running time of "slices" is of course slower, because, while every slice is very fast (even faster if we use similar slices, that is, with close value of a, as starting points while running them for 8-9 iterations and using the result as a starting point for the other slices), about 0.2ms, which must be multiplied by the number of slices. On the pro side, the slices can be trivially parallelized on multicore machines. For example, for 40 slices with 8 cores, the running time is approximately 40*0.25/8 = 1.25 ms.

4 Results

In order to test the proposed solution, we ran it on a large set of inputs provided by Yuval Alfassi. This data consisted of 10,000 sents of pairs of lines and points: each set has 50 lines/points pairs. The small number of lines and points means that the time spent for processing them is much

smaller than the time required for solving the optimization problem, which was also sent to us by Yuval Alfassi. A comparison of the runing times is provided in the following figure. The optimization results were exactly the same (up to six digits accuracy), and the improvement of the proposes method in running times was by about a factor of 10, which is due to the Hessians being much smaller (34 X 34 instead of 70 X 70), the fact that the "extra" 15 variables of the quadratic polynomial $q_2(x, y, z, w)$ are not required, and that the matrices of the dual problem we solve (presented here is pages 5-7) are very sparse.

As described before, the "slices" method is slower and slightly less accurate, however it has the following advantage:

- It is easily parallelizable.
- It provides a solution which is guaranteed to approximate the correct solution.

Note: The X axis represents average runtime in milliseconds

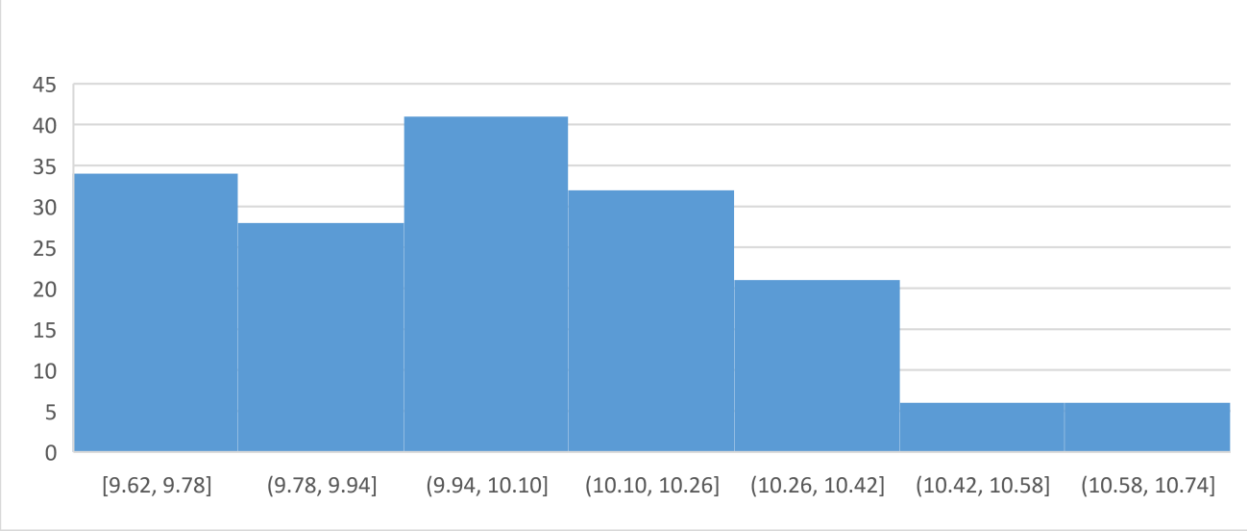

Figure 1: Result for the method presented in the IROS 2022 conference (Ibrahim Jubran, Fares Fares, Yuval Alfassi, Firas Ayoub, Dan Feldman: Newton-PnP: Real-time Visual Navigation for Autonomous Toy-Drones.)

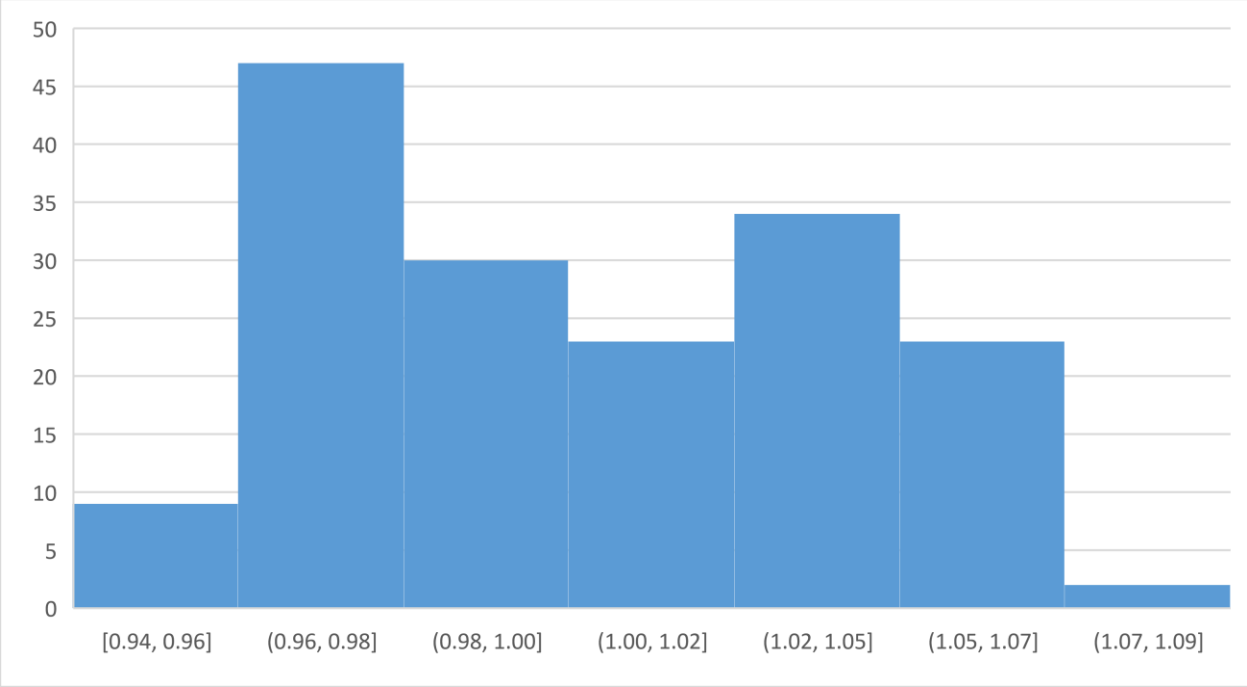

Figure 2: Result for proposed method