Camera Pose and Focal Length Estimation Using Regularized Distance Constraints

Ekaterina Kanaeva[1]
kanaeva.katerina6@gmail.com
Lev Gurevich[2]
lev@divisionlabs.com
Alexander Vakhitov[1]
a.vakhitov@spbu.ru

[1] Chair of Software Engineering
St. Petersburg State University
St. Petershurg, Russia

[2] Digital Vision Labs LLC
196602 Sapernaya 22-18,
St. Petersburg, Pushkin, Russia

Abstract

We propose a new method for camera pose estimation with unknown focal length (PnPf problem). We combine projection equations and distance constraints in a single statistically significant cost function in the form of least squares. We fix the space of the search as a linear combination of several right singular vectors of the least squares system matrix. We use linear programming techniques to find feasible solutions faster. Then we do nonlinear refinement with Levenberg-Marquardt. Numerical experiments demonstrated that the method is faster than the state-of-the-art methods for point numbers up to several hundreds, and real-life structure-from-motion demonstrates its applicability for models having 10^3-10^5 points. It has the same accuracy of estimates as the the state-of-the-art methods. We show that the method offers a tradeoff between speed and accuracy, allowing the estimation to run several times faster while slightly increasing the mean reprojection error.

1 Introduction

Camera pose and intrinsic parameters estimation from n 2D-to-3D point correspondences is a known problem in computer vision and photogrammetry. Possible applications are reconstruction of 3D scenes from large unordered internet photo collections [1], augmented reality [5] etc. Many modern consumer systems (smartphones) currently have autofocus, and usually do not provide any interface to measure the current focus length, and EXIF information can be absent or imprecise.

Depending on the set of unknown parameters, the problem is called Perspective-n-Point (PnP) when only absolute camera pose is unknown, PnPf when focal length is unknown as well or PnPfd when additionally distortion coefficient is unknown. Projection error functions are highly non-convex in focal length, so before methods for PnPf were published, the only choice was to do exhaustive search not suitable for real-time applications, for embedded and mobile systems, or in cases when millions of images should be processed.

A special case is when the number of point correspondences n is minimal required to find a unique set of unknown parameters. Such problems are known as P3P, P4Pf, P4Pfd [2, 3]. The methods solving these problems are fast and usually used in RANSAC loop

[6]. However significant noise in image projections or uneven points' distribution in space leads to a situation when minimal methods provide far from optimal solutions which cannot be easily improved. As noted in [12], percentage of outliers can be such that 6-point and 4-point methods will need similar amount of iterations by theoretical bound, but 6-point method always gives more reliable estimations. In [9] applicability of EPnP-like methods with RANSAC was demonstrated.

In any case, if the system has no false point detections, such as one based on marker detector [5], or if the system uses RANSAC, a PnP (PnPf, PnPfd) method gives a final solution from arbitrary amount of points.

This paper is devoted to a method for PnPf problem for arbitrary amount of points. We consider both planar and non-planar cases. Planar case is practically important for augmented reality [5]. General case is used in all structure from motion problems, for example [1].

There are well known solutions to the PnP problem with arbitrary amount of points (EPnP [9], RPnP [10], OPnP [15]). The main requirement was to make the methods' runtime low for hundreds of points. We describe EPnP further because our method builds on it. RPnP divides point set into triplets and derives constraints for these triplets using known angles and distances between points [10].

The EPnP was extended to PnPf problem in [14], we refer to this method as UPnPf. RPnP inspired the authors of [16] to propose a method GPnPf+GN for PnPf problem. They use angle constraints to build specific polynomial system and solve it, then they use nonlinear refinement with Gauss-Newton algorithm. It gave superior results to [14] both in speed and accuracy in general case, and was more accurate in planar case, although UPnPf [14] was faster.

The method proposed here is based on EPnP. It is 2-3 times faster than GPnPf+GN for amount of points up to several hundreds. It has similar accuracy to GPnPf+GN and is more than 2 times faster. It is flexible in a way that different sets of parameters allow the end user to control the accuracy/speed tradeoff.

In the following sections we review the EPnP method, formulate the theoretical foundations of the proposed algorithm (section 2), explain the algorithm and implementation details (section 3), give results of synthetic and real experiments (section 4).

2 Preliminaries

We have 3D points in model frame $\{\mathbf{p}_i^m\}_{i=1}^N$, $\mathbf{p}_i^m \in R^3$, they are transformed to points in camera frame $\{\mathbf{p}_i^c\}_{i=1}^N$ by an euclidean transform and then projected onto a camera, their projections are $\{(u_i, v_i)\}_{i=1}^N$. The camera has focal length f. Image coordinates origin is in the principal point, camera has unit aspect ratio and zero skew [7].

We review the approach used by EPnP [9]. We address the general case, and for the planar case analogous formulas hold with 3 basis points instead of 4 (see below).

2.1 The EPnP Method

The barycentric representation of 3D points allows to express n 3D points $\mathbf{p}_i$ as a frame-independent linear combination of 4 basis points $\mathbf{c}_j$:

$$\mathbf{p}_i = \sum_{j=1}^{4} \alpha_i^{(j)} \mathbf{c}_j, \;\; i = 1 \ldots n, \;\; \sum_{j=1}^{4} \alpha_i^{(j)} = 1.$$

We concatenate the basis points' coordinates in model frame $\mathbf{c}_j^m$ in a matrix $\mathbf{C}^m$, points' 3D coordinates $\mathbf{p}_i^m$ in a matrix $\mathbf{P}^m$, coefficients vectors $\mathbf{a}_i = (\alpha_i^{(1)}\, \alpha_i^{(2)}\, \alpha_i^{(3)}\, \alpha_i^{(4)})^T$ in a matrix $\mathbf{A}$. Then the previous equations for all the i's turn into a single matrix equation $\mathbf{P}^m = \mathbf{C}^m\mathbf{A}$. In camera frame, we denote point coordinates as $\mathbf{P}^c$, basis points coordinates as $\mathbf{C}^c$, and the equation is $\mathbf{P}^c = \mathbf{C}^c\mathbf{A}$. The PnP problem is equivalent to finding $\mathbf{C}^c$ [14]. In case of coplanar points $\mathbf{p}_i$, we need only 3 basis points.

Each point projection as described in [9, 14] leads to two independent linear equations. Denote the rows of matrix $\mathbf{C}^c$ as $\mathbf{g}_1$, $\mathbf{g}_2$ and $\mathbf{g}_3$, the point $\mathbf{p}_i^c$ coordinates as $(x_i\, y_i\, z_i)^T$, denote $\mathbf{d}_1 = f\mathbf{g}_1$, $\mathbf{d}_2 = f\mathbf{g}_2$, then the projection equations are:

$$\begin{cases} \mathbf{g}_3\mathbf{a}_i u_i = \mathbf{d}_1\mathbf{a}_i \\ \mathbf{g}_3\mathbf{a}_i v_i = \mathbf{d}_2\mathbf{a}_i \end{cases}.$$

All projection equations form a linear system with matrix $\mathbf{M}$ of size $n \times 12$:

$$\mathbf{M}\mathbf{x} = 0, \quad \mathbf{x} = \begin{pmatrix} \mathbf{d}_1 & \mathbf{d}_2 & \mathbf{g}_3 \end{pmatrix}^T. \tag{1}$$

The solution lies in a null-space (kernel) of M and can be expressed as a linear combination of kernel basis with coefficients β_i

$$\mathbf{x} = \sum_{i=1}^{N} \beta_i \mathbf{q}_i, \tag{2}$$

where $\mathbf{q}_i$ are the right-singular vectors of $\mathbf{M}$ corresponding to the N null singular values of $\mathbf{M}$.

When we have noisy projections, for $n \geq 6$ none of the singular values of M is exactly 0, but several are very small. We choose N to be the dimension of the subspace corresponding to the right-singular vectors which we will call approximate kernel of $\mathbf{M}$, or $AK(\mathbf{M})$. The problem is to find a needed element in $AK(\mathbf{M})$. The elements of AK correspond to projective transforms satisfying (1), and we need to choose the euclidean one among them. Theoretically, euclidean transform of space is the only one which does not change the distances between points:

$$\|\mathbf{c}_i^c - \mathbf{c}_j^c\|^2 = \|\mathbf{c}_i^m - \mathbf{c}_j^m\|^2 = r_{ij}^2, \tag{3}$$

where r_{ij} is a known distance between i-th and j-th basis points. There are 6 constraints in general case and 3 in planar case. Distance constraints are quadratic w.r.t. β_i and in the same time are linear w.r.t. $\mathbf{b}$:

$$\mathbf{b} = \begin{pmatrix} \beta_1^2 & \ldots & \beta_N^2 & \beta_1\beta_2 & \ldots & \beta_{N-1}\beta_N \end{pmatrix}^T. \tag{4}$$

Therefore we get a linear system of distance constraints with vector $\mathbf{b}$ of unknowns. We denote the matrix for this system as $\mathbf{L}$ and the distances vector as r, then the system is

$$\mathbf{L}\mathbf{b} = r. \tag{5}$$

The EPnP method's last step is to find a solution of the distance constraints system in case of $N = 1,\dots,4$. If $N \geq 3$, in PnP problem linearized distance constraints do not give enough equations to solve the problem, so in EPnP additional constraints are formed using a technique known as relinearization [8]. We have different approach to solving (5).

3 Our Method for PnPf Problem

3.1 Regularization of the Distance Constraints for PnP

In case of noise in projections, the system (1) will not hold exactly. If we solve it in the least squares sense, we minimize $F_1(\mathbf{x}) = \|\mathbf{M}\mathbf{x}\|^2 \to \min$. Least squares solution is as well the ML (maximum likelihood) solution if the depths (z_i) are the same and camera is general projective. If depths are different then to obtain ML solution we need to weight the projection equations with weights $1/z_i$ as explained in [7]which we do not do in this paper. Camera considered here is euclidean, but F_1 describes only projective constraints, so to choose the suitable solution we need some additional cost function, which is described next. Currently, let us look closer at the structure of F_1.

If we substitute to F_1 an element of AK given by (2), then it will be equal to $F_1(\mathbf{x}) = F_1(\sum_{i=1}^{N} \beta_i \mathbf{q}_i) = \sum \beta_i^2 \sigma_i^2$, where σ_i is the singular value of M corresponding to the vector q_i.

Next, let us return to the problem of choosing the euclidean transform among projective ones. The system (5) is defining constraints for euclidean transform. It can be solved in the least squares sense: $F_2(\mathbf{x}) = \|\mathbf{L}\mathbf{b} - \mathbf{r}\|^2 \to \min$.

We propose to regularize F_1 with F_2, that is, to consider a sum of two cost functions. First one corresponds to minimization of reprojection error with some projective transform. The other one is about bringing this projective transform as close as possible to an euclidean one. We form a cost function $F_R(\mathbf{x}) = F_2(\mathbf{x}) + \gamma F_1(\mathbf{x})$, where γ is a regularization coefficient.

The problem to minimize F_R is equivalent to solving in least squares sense the augmented system. We add to the original $6 \times D$ matrix $\mathbf{L}$ the diagonal matrix $\mathbf{L}_R^1$ of size $N \times D$ with corresponding singular values of $\mathbf{M}$:

$$\mathbf{L}_R^1 = (diag(\sigma_1,\dots,\sigma_N)\ \mathbf{0}).$$

Finally we get the following *regularized distance constraints system* (RDC system)):

$$\hat{\mathbf{L}}\mathbf{b} = \hat{\mathbf{r}}, \quad \hat{\mathbf{L}} = \begin{pmatrix} \mathbf{L} \\ \mathbf{L}_R^1 \end{pmatrix}, \quad \hat{\mathbf{r}} = \begin{pmatrix} \mathbf{r} \\ \mathbf{0} \end{pmatrix}. \tag{6}$$

Our preliminary experiments have shown that regularization can be used instead of relinearization [8] proposed in the original EPnP paper [9] in classical PnP problem.

3.2 Regularized Constraints System for PnPf

Distance constraints for the PnPf problem (analogously to (**??**) for PnP) have the form:

$$r_{ij}^2 = \|\mathbf{c}_i - \mathbf{c}_j\|^2 = \sum_{k=1}^{3} \|\mathbf{c}_i^{(k)} - \mathbf{c}_j^{(k)}\|^2 = \frac{1}{f^2}((\mathbf{d}_i^{(1)} - \mathbf{d}_j^{(1)})^2 + (\mathbf{d}_i^{(2)} - \mathbf{d}_j^{(2)})^2) + (\mathbf{g}_i^{(3)} - \mathbf{g}_j^{(3)})^2. \tag{7}$$

This is a quadratic constraint in $\mathbf{d}_i, \mathbf{g}_i$, but if we choose the unknowns vector $\mathbf{b}$ (analogously to (4)) another way, it becomes linear with $\mathbf{b^1}$ equal to $\mathbf{b}$ given in (4) and:

$$\mathbf{b^2} = f^2\mathbf{b^1}, \quad \mathbf{b}^T = (\mathbf{b^1}^T, \mathbf{b^2}^T)^T. \tag{8}$$

We can define $F_1^f(\mathbf{x})$, $F_2^f(\mathbf{x})$ and $F_R^f(\mathbf{x})$ for PnPf analogously to $F_1(\mathbf{x}), F_2(\mathbf{x})$ and $F_R(x)$ for PnP, but they will depend not only on β_i, $\mathbf{b}$, but also on $f^2\mathbf{b}$, $f\beta_i$ for $i = 1 \dots N$.

4 The Algorithm Formulation

4.1 Candidate Solutions from Approximate Kernel

We explain in details the procedure to find candidate solutions $\{R_i, t_i, f_i\}$ having a fixed $AK(\mathbf{L}_S)$ of dimension k. If $b \in AK(\mathbf{L}_S)$, then $\mathbf{b} = \mathbf{b_0} + \mathbf{F}\mathbf{w}$, where $\mathbf{w} \in R^k$. Let us split the vectors $\mathbf{b}$, $\mathbf{b_0}$ and a matrix $\mathbf{F}$ in two halves $\mathbf{b^1}, \mathbf{b^2}$, $\mathbf{b_0^1}$, $\mathbf{b_0^2}$ and $\mathbf{F^1}, \mathbf{F^2}$ such that $\mathbf{b^i} = \mathbf{b_0}^i + \mathbf{F^i}\mathbf{w}$, $i = 1, 2$. as We formulate a relaxed collinearity constraint: let us find such $\mathbf{w}$ that

$$\langle \mathbf{b^1}, \mathbf{b^2} \rangle = \langle \mathbf{b_0}^1 + \mathbf{F^1}\mathbf{w}, \mathbf{b_0}^2 + \mathbf{F^2}\mathbf{w} \rangle > 0. \tag{9}$$

We have as well a positivity constraint on elements of $\mathbf{b}$ (derived from (4)):

$$\exists i = 1 \dots N \; b^{(i)} > 0 \;\wedge\; b^{(D+i)} > 0. \tag{10}$$

As you see, we relax (4) so that $\mathbf{w}$ is a candidate solution if (10) is satisfied at least for some index i and (9) holds. Next we show that we can find a candidate solution by linear programming using MATLAB's *linprog*. Constraint (10) is linear, but (9) is quadratic, so we substitute it with a stronger system of linear inequalities.

In particular, let us denote as $\mathbf{T}$ the orthogonal coordinate transform such that $\mathbf{w}' = \mathbf{T}\mathbf{w}$ and matrix of a quadratic form in constraint (9) becomes diagonalized, so (9) is transformed to:

$$w'^T diag(j_1, \dots, j_k) w' + h > 0. \tag{11}$$

If j_i is positive, then for sufficiently large $|w'^{(i)}|$ if all the other components of w' except the i-th are 0, the inequality will hold. In particular, we get a constraint pair:

$$w'^{(i)} > \sqrt{e_i} \vee w'^{(i)} < -\sqrt{e_i}, \; e_i = -h^{(i)}/j_i. \tag{12}$$

To generate candidate solutions, we loop over i and use *linprog* to find $\mathbf{w}$ satisfying both (10) for this i and either positive or negative version of (12).

4.2 Main Algorithm [1]

$\mathbf{L}_S$ is generated using coefficient γ, $\gamma = 10^{-5}$ in general case and $\gamma = 0$ in coplanar case.

We do several iterations, with index variable $N = 1 \dots S$. On each iteration, $AK(L_S)$ of size N is found and candidate solutions from it are generated as described before. For a candidate solution we then check several conditions:

1. corresponding estimate of focal length f is in the interval $[30, 20000]$;

[1] The implementation of the algorithm can be accessed at http://sites.google.com/site/alexandervakhitov/projects/epnpfr.

Parameter	Fast Mode	Full Mode
Initial damping factor	10^{-3}	10^{-1}
Stopping threshold for $\|\|J^T e\|\|_{inf}$	10^{-5}	10^{-10}
Stopping threshold for $\|\|Dp\|\|_2$	10^{-5}	10^{-10}
Stopping threshold for $\|\|e\|\|_2$	10^{-5}	10^{-10}
Finite difference approximation step	10^{-6}	10^{-6}

Table 1: Levenberg-Marquardt Parameters.

2. the average reprojection error for points is compared to the expected error e (chosen to be 7.5 pixels in all experiments unless stated otherwise).

If a solution passes these tests, it is stored in a collection.

After a loop over N, we choose the solutions which have the least amount of points lying behind the camera, and among these solutions choose the by comparing the value of $F_R^f(\mathbf{x})$. This chosen solution is subsequently refined by Levenberg-Marquardt procedure, the parameters of which can be found in the table (1) (see their description in [12]).

4.3 Improvement of Runtime for Large Point Counts

As it was observed in [16], the time of EPnP-based methods grows linearly with respect to point number. Most of the time for large point numbers is consumed by construction of matrix $\mathbf{M}$ (1) and SVD of it.

First of all, instead of computing SVD of $\mathbf{M}$ we compute SVD of $\mathbf{M}^T\mathbf{M}$. This matrix has same (up to sign) right singular vectors as $\mathbf{M}$, however its size is just 12×12 (or 9×9 in planar case). We will construct the system matrix $\mathbf{M_{red}}$ equivalent to $\mathbf{M}$ which can be computed more efficiently. We remind that structure of $\mathbf{M}$ can be described with the use of matrix $\mathbf{A}$ defined in section 2.4.

Let us denote $\mathbf{U} = diag(u_1,\dots,u_n)$, $\mathbf{V} = diag(v_1,\dots,v_n)$. Then denote $\mathbf{U_1} = -\mathbf{U}\mathbf{A}^T$, $\mathbf{V_1} = -\mathbf{V}\mathbf{A}^T$. We can rearrange the unknowns in the system $\mathbf{M}\mathbf{x} = 0$ so that we get the system $\mathbf{M_{red}}\mathbf{x} = 0$, and

$$\mathbf{M_{red}} = \begin{pmatrix} \mathbf{A}^T & \mathbf{0} & \mathbf{U}_1 \\ \mathbf{0} & \mathbf{A}^T & \mathbf{V}_1 \end{pmatrix}.$$

Then computation of $\mathbf{M}_r^T\mathbf{M}_r$ can be done in blockwise manner. We propose to do the rearranging, compute $SVD(\mathbf{M_{red}}^T\mathbf{M_{red}})$ and then rearrange back the components of the right singular vectors. These equivalent formula transformations significantly improve the computation time for large amounts of points as can be seen in the following section.

5 Experiments

We made two sets of experiments: comparative test based on the toolkit developed for the paper [16] and a test using real images.Each of the state-of-the-art methods are formulated in two ways: with and without nonlinear refinement. We aim at higher quality, so for the general case we chose the methods with refinement UPnPf+GN [14], GPnPf+GN [16]. For planar case, we chose for comparison UPnPf, because UPnPf+GN works only in general case, and GPnPf+GN. The RPnP method [10] is chosen as a state-of-the-art PnP method to

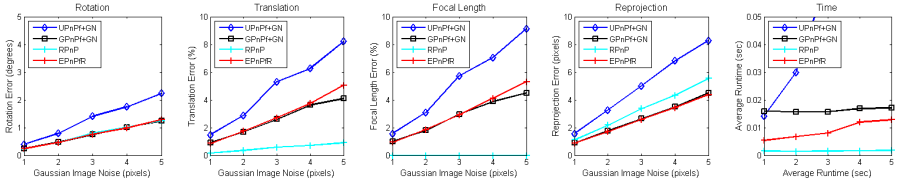

Figure 1: Median estimation errors for camera pose and focal length, points in general 3D configuration, mean reprojection error and time of the PnPf methods w.r.t. varying noise level

check how unknown focal length influences the accuracy of estimates. This method is run with true focal length values. We denote our method as EPnPfR. We run the experiments on a notebook with 8 GB RAM and 2.5 GHz Intel Core i5-4200M processor.

5.1 Comparative Synthetic Experiments

We model a camera with principal point at the origin of pixel coordinates. The image resolution is 800×600. The focal length is sampled from uniform distribution from 200 to 2200. The points are randomly generated in the box $[-2,2] \times [-2,2] \times [4,8]$, the point number is n. The points are projected onto the camera and then noise is added. The noise is normally distributed and zero-centered with variance δ.

We measure translation and focal length errors relative to the true value, other errors are in absolute units (pixels or degrees). We measure runtime in seconds.

5.1.1 Behavior with Varying Noise Level

In the first experiment, we change the noise std. dev. from 1 to 5 with step 1. We take 6 random points and do 500 trials for every noise variance. At the fig. 1 we show the median error over all the experiments for each method and noise level considered. Proposed method delivers almost the same quality of estimates as GPnPf+GN but is 1.5-2 times faster. UPnPf+GN is slower and less accurate. We compare the results with RPnP and see that unknown focal length results in errors in translation estimates, while rotation is not influenced (also observed in [16]).

In planar case (see fig. 3) we see that while UPnPf outperforms others in terms of speed, but offers lower accuracy of estimates, our method gives accuracy comparative to GPnP+GN while performing 2 times faster than GPnPf+GN.

At the fig. 2 you see the comparison of the variance of errors for best methods, GPnPf+GN and the proposed one (EPnPfR). The variance is higher in planar case. In fact both methods in planar case seldom give outlying results with huge reprojection errors. As for our method, usually the reason is that the estimated focal length does not lie in the prescribed interval explained in the previous section.

5.1.2 Accuracy w.r.t. Varying Point Number

We fix $\delta = 2$ and vary n from 6 to 15. At the fig. 4 you can see the reprojection errors for the methods (output of MATLAB boxplot function). We observe almost the same behavior

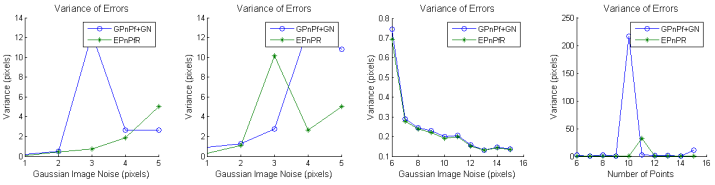

Figure 2: Variance of reprojection errors for GPnPf+GN and proposed method for (left to right): general and coplanar configurations w.r.t. noise leve, general and coplanar configurations w.r.t. point number.

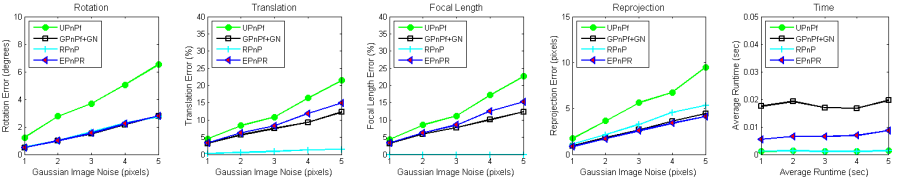

Figure 3: Median estimation errors for camera pose and focal length, mean reprojection error and time of the PnPf methods w.r.t. varying noise level.

of GPnPf+GN and EPnPfR. The latter is more than 2 times faster. As shown at the fig. 2, error variance does not show presence of big outliers, and for the both compared methods is very similar.

In planar case (see fig. 5), we see that our method EPnPfR outperforms other ones in terms of translation and focal length errors. The speed is more than 2 times faster. At the fig. 2 you see that seldom big estimation errors happen.

5.1.3 Runtime w.r.t. Varying Point Number

To analyze the runtime for large point counts, we fix the noise variance $\delta = 2$ and run the methods for $n = 10, 110, \ldots, 910$.

We run our method in two configurations: standard one and fast one (EPnPfRFast). The only difference is in Levenberg-Marquardt method parameters (see table 1). The reprojection error for GPnPf+GN, EPnPfR and EPnPfRFast for point numbers greater than 100 is very

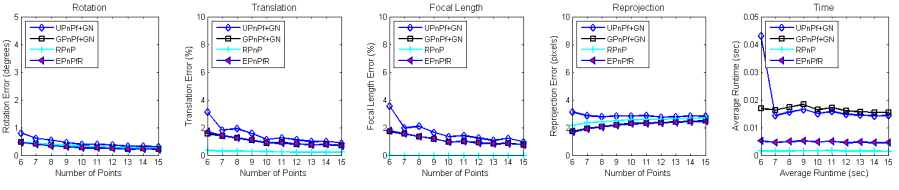

Figure 4: Median estimation errors for camera pose and focal length, planar points, reprojection error and time of the methods w.r.t. varying point number for points in general configuration.

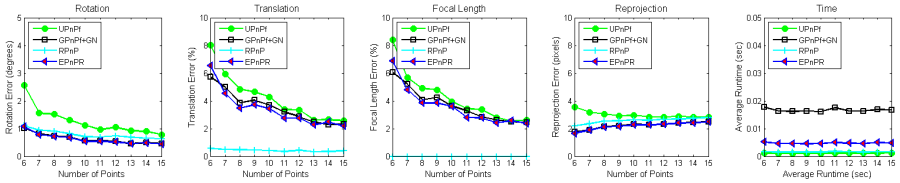

Figure 5: Median estimation errors for camera pose and focal length, planar points, mean reprojection error and time of the PnPf methods w.r.t. varying point number for coplanar points.

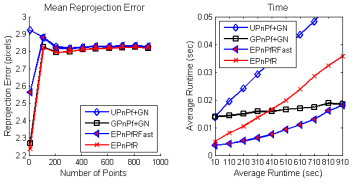

Figure 6: Mean reprojection error and runtime for large point numbers.

similar. The runtime of EPnP-based methods grows linearly. The EPnPfRFast is much faster than GPnPf+GN. We see that most of the time in EPnPfR for large point numbers is spent in nonlinear refinement.

5.2 Real Experiments

The aim is to demonstrate applicability of the algorithm in a real setting. We made three shots of a non-moving scene with Nikon D3100 camera with several focal length settings. We have chosen one pair of frames with focal length 35 mm (initial pair) for the model reconstruction. We performed SIFT [13] matching of points between the frames of the pair. We estimated fundamental matrix using 8-point algorithm [7], found the consensus point set using RANSAC [6]. Using the focal length estimated from the EXIF information we found camera matrices, triangulated points and did euclidean bundle adjustment using sba package [11] as described in [7]. Finally, the model was scaled using the 0.24 m distance measured in the scene.

We matched the SIFT points from one of the frames in the initial pair and every other frame. We ran our algorithm with the RANSAC loop, choosing 6 points [7]. When both methods returned results, they were different less than for 1% in focal length and reprojection error, except 105 mm focal length (average error GPnPf+GN 2.53 pix, EPnPfR 0.77 pix).

We see that the EPnPfR is more robust to model inaccuracy, and it can be successfully used in RANSAC loop although was not designed for this purpose.

The following table lists some of the results of the PnPf experiments. We have taken SIFT points from just one picture for matching, so depending on the viewpoint we have experiments with 100-300 as well as 1000-2695 correspondences. The focal lengths in mm are 18,24,35,50,75,105, and we have calculated them in pixels using EXIF data and Nikon-provided sensor size.

When we write -1 in the column of the GPnPf+GN method, we mean that the error in

Figure 7: Results of PnPf for focal length of 18, 24, 50, 75 and 105 mm. Model is built with focal length 35 mm.

True focal length, pix	Point number	EPnPfR	GPnPf+GN
1795.37	125	1.04 (1781.42)	1.04 (1781.64)
1795.37	1091	0.78 (1775.86)	0.77 (1774.70)
1795.37	1324	1.50 (1816.60)	1.50 (1814.40)
2393.83	101	1.13 (2303.53)	1.13 (2301.69)
2393.83	300	0.62 (2337.47)	0.62 (2337.19)
2393.83	1206	1.01 (2321.59)	1.00 (2320.16)
3491.00	114	1.29 (3454.11)	-1.00 (-1.00)
3491.00	2688	0.13 (3490.66)	-1.00 (-1.00)
3491.00	2695	0.14 (3490.96)	0.13 (3491.00)
4987.14	407	1.14 (5204.69)	1.13 (5200.91)
4987.14	1058	0.78 (5338.28)	0.78 (5340.13)
4987.14	1916	0.63 (5500.42)	-1.00 (-1.00)
7480.71	444	1.14 (8096.16)	1.14 (8083.26)
7480.71	1037	1.17 (9078.29)	1.17 (9079.25)
7480.71	1743	1.10 (8338.50)	1.11 (8341.89)
10473.00	369	1.25 (15184.75)	-1.00 (-1.00)
10473.00	1169	0.94 (14790.30)	-1.00 (-1.00)
10473.00	1019	0.77 (17843.48)	2.53 (6503.70)

Table 2: The true and estimated focal length and reprojection errors for 18 experiments with real images

the script happened. This error leads to NaN arrays and failure of the script. We just made an error returning operator in the script in this case.

6 Conclusions

We have derived a novel fast method for the PnPf problem. It is based on EPnP [9] but we reduce the search space greately by introducing the regularization. We show how a preprocessing to reduce the computation time of EPnP-based methods for large point counts, which was mentioned as a disadvantage of these methods in [16].

We show that the method is capable of fast having state-of-the-art accuracy. On a real dataset, we see that the method we propose is more robust to 3D model inaccuracy because it behaves similarly to [16] on synthetic data but works on real datasets where [16] does not.

We'd like to thank the anonymous reviewers for their attention, detailed analysis and valuable critics. It contributed a lot to the paper.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[3] M. Bujnak, Z. Kukelova, and T. Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *Computer Vision–ACCV 2010*, pages 11–24. Springer, 2011.

[4] Luis Ferraz, Xavier Binefa, and Francesc Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 501–508. IEEE, 2014.

[5] M. Fiala. Artag, a fiducial marker system using digital techniques. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 590–596. IEEE, 2005.

[6] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[7] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[8] A. Kipnis and A. Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In *Advances in cryptologyâĂŤCRYPTOâĂŹ99*, pages 19–30. Springer, 1999.

[9] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[10] S. Li, C. Xu, and M. Xie. A robust o (n) solution to the perspective-n-point problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1444–1450, 2012.

[11] M.I. A. Lourakis and A.A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009. doi: http://doi.acm.org/10.1145/1486525.1486527.

[12] M.I.A. Lourakis. levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++. [web page] `http://www.ics.forth.gr/~lourakis/levmar/`, Jul. 2004. [Accessed on 31 Jan. 2005.].

[13] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[14] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer. Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):2387–2400, 2013.

[15] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2344–2351. IEEE, 2013.

[16] Y. Zheng, S. Sugimoto, I. Sato, and M. Okutomi. A general and simple method for camera pose and focal length determination. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 430–437. IEEE, 2014.