sensors

Article

Visual Localization Based on Torus-like Surfaces

Xuandong Liu [1], Lihong Luo [1,*] and Bingren Shen [2]

[1] School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China; 13828417894@163.com
[2] School of Digital Media, Guangdong University of Technology, Guangzhou 510006, China; shenbingren0@163.com
* Correspondence: luolhgdut@163.com

Abstract: Previous visual localization started from point correspondences (PCs) to estimate poses. This article takes the camera position as the entry point and finds that the camera position solution set rotates around an axis connected by two observed 3D points to form a surface called a torus-like surface (TLS). The relevant parameters of TLS are calculated based on PCs and camera intrinsic parameters. In order to reduce the number of solutions, this article uses four PCs to construct three TLSs. By utilizing four PCs, the pose determination problem is reformulated as the task of finding the optimal intersection of three surfaces. By using the step-size adaptive tracking method, the candidate set of intersections can be quickly and accurately found. Combining the feature information of intersections on TLS and the camera intrinsic parameters, the optimal position is obtained. Based on this position, the rotation matrix can be determined. In the synthetic data experiments and the dataset experiments based on image localization, it is shown that the visual localization based on TLS is more accurate than current state-of-the-art methods, which provides a new entry angle and effective ideas for visual localization. Its accuracy and practicality are fully demonstrated in the application test of augmented reality indoor navigation.

Keywords: visual localization; point correspondence; surface intersection; tracking method; augmented reality; indoor navigation

check for updates

Citation: Liu, X.; Luo, L.; Shen, B. Visual Localization Based on Torus-like Surfaces. *Sensors* **2024**, *24*, 7349. https://doi.org/10.3390/s24227349

Academic Editor: Robert Sitnik

Received: 19 October 2024
Revised: 7 November 2024
Accepted: 17 November 2024
Published: 18 November 2024

1. Introduction

Mobile devices typically use inertial measurement units within the device for inertial localization navigation. However, inertial navigation algorithms have some problems, such as mismatch and inconsistent image features due to changes in camera position while the device is moving, which affects the accuracy and continuity of localization [1]. In addition, it may be disturbed by its own temperature, zero bias, vibration, and other factors to produce errors [2]. These errors will gradually accumulate during the moving process, affecting the device's accurate cognition of its own position, which in turn leads to the deviation of the navigation path. Therefore, it is necessary to adopt correction algorithms. The most common correction method is image-based visual localization [3,4].

The dominant approach to image-based visual localization is to estimate the camera pose by querying the correspondences between the 2D features of the image and the known 3D points in the scene. These correspondences are likely to have mismatched information and thus need to be filtered with Random Sample Consensus (RANSAC) [5] or its derivatives [6] to find suitable correspondences. The most representative mainstream methods of this kind are P3P [7] and Infinitesimal Plane-based Pose Estimation (IPPE) [8]. P3P uses three PCs to estimate the camera position, but there is a dangerous cylinder problem [9] that makes the solution unstable, while IPPE requires four coplanar PCs to estimate the pose, which is accurate and stable but has more stringent requirements for coplanar 3D points.

In order to improve the accuracy and speed of absolute pose estimation, people have started to investigate feature information other than PCs. For example, local affine correspondences are obtained by combining 2D matching [10], or external pose information is introduced from the inertial measurement unit (IMU) [11], and the pose is estimated by single-point affine correspondences and an estimate of the surface normal at the 3D point [12].

In order to balance real-time estimation and accuracy, people are more likely to improve real-time estimation by reducing PCs while introducing other feature information for accuracy compensation. Therefore, we started to explore whether there is a method to maintain high real-time performance without reducing PCs but still solve the accurate position, and at the same time, the method does not impose harsh restrictions on the 3D points. To this end, we try to find the intrinsic connection between PCs and revisit the monocular imaging model, starting from the camera's trajectory for the 3D points, and find that the camera trajectory forms a torus-like surface model at the two visible 3D points, and its view cone is always perpendicular to the tangent plane of the torus-like surface at the camera's position, and by combining the camera's internal references with the 3D points, we can easily obtain the 3D coordinates of the projection points that are projected onto the imaging surface. Following this, the rotation matrix is determined.

Compared with other methods, our idea is different. When the reference camera and the query camera observe the same four 3D points, the viewing angle will be adjusted following the different positions to ensure that the same four points are observed, three torus-like surfaces are constructed based on these four 3D points, and the optimal intersection of three torus-like surfaces will be used as the position solution for the camera, and then the rotation matrix is determined based on 2D-3D correspondences, camera intrinsics, and the information of the estimated position. This method estimates the position more accurately, and we demonstrate this through experiments on synthetic and real data.

Our contributions are as follows:

- We propose a novel method for solving the pose, which converts the problem of solving the pose with four PCs that are not colinear between three 3D points into a problem of intersecting surfaces to find intersections. We apply a tracking algorithm with an adaptive step size to determine surface intersections, which are stable and fast. Combining the intersection positions, the PCs, and camera intrinsics, the rotation matrix will be estimated;
- Experiments on synthetic data demonstrate the numerical stability of our approach for solving and analyzing its performance under various types of noise;
- Experiments on two benchmark datasets based on image localization show that our method can estimate more accurate camera pose than state-of-the-art methods in comparable time;
- Application testing of TLS visual localization applied to augmented reality in-door navigation to verify its utility and accuracy.

2. Related Work

Perspective-n-Points (PnP) is the problem of solving the camera pose by using n pairs of matching world and image coordinates when the object's coordinates in the world coordinate system and its image coordinates are known. Previous research has obtained the following patterns after deduction: when n is 1 or 2, there are infinite solutions; when n is 3, there are at most four solutions; when n is 4 and four points in the world coordinate are coplanar, there is only one solution, etc. [13,14].

The PC-based PnP problem has a minimum n value of 3, which is also known as the P3P problem. The problem of estimating the absolute pose of a calibrated camera from three PCs has been considered for almost two centuries. Recent research has continued to improve the speed and stability of P3P solutions [15,16], including an updated method recently introduced [17].

In the past decade, many methods have been proposed to solve the PnP problem for n > 3. These methods can be roughly categorized into iterative and non-iterative.

Classical iterative methods formulate the PnP problem as an optimization objective function through the relationship between the 3D control points and the 2D projection points and then use an iterative optimization method such as Levenberg–Marquardt [18] or L-BFGS [19] to estimate the pose. For example, in [20,21], such methods are prone to fall into local minima without a suitable starting point, which can lead to degradation of accuracy.

A non-iterative approach looks for constraint information among the PCs to estimate the pose. Collins and Bartoli derived new constraints and minimum solutions for the absolute pose of the camera given four coplanar PCs by selecting an optimal point on a plane to compute a first-order approximation to the homography (an affine transformation) and then computed the camera's pose based on the 2D point projections and the first-order approximation local transformations. Mills [22] describes a four-point solver for fundamental matrix estimation, while Barath [23] develops a five-point solver for estimating the fundamental matrix from oriented features.

Recently, constraint information other than PCs has been utilized to estimate the pose. Köser and Koch introduced a solution for absolute pose estimation from a single affine correspondence (AC) in orthogonal reference images [10], while Ventura et al. developed a generalized absolute pose solution from perspective references based on the knowledge of the surface normal. In order to reduce the minimum sample size of the absolute pose, [24] proposed introducing external attitude information from an inertial measurement unit (IMU) to solve the pose.

The above methods start from PCs or combine constraints other than PCs to estimate pose but do not start from the pose solution set to explore new constraints to estimate pose. We review the perspective geometric model, materialize the camera intrinsics and focal length as a view cone, explore what kind of motion trajectory the vertices of this view cone follow to determine the solution set of the camera position, and then find the exact pose solution.

3. Methods and Works

P and Q are two points in world coordinate, $OKLMN$ is the view cone, point O is the camera position, and p and q are the projections of P and Q on imaging plane $KLMN$. Combining points P, Q, and O, we can define a plane PQO.

Under the condition that the view cone is guaranteed to be invariant (i.e., camera intrinsics are invariant), the trajectory of the solution set formed by the camera position O on the plane OPQ is shown in Figure 1. The trajectory is known to be a circular line, and the imaging angle ξ is unchanged when the view cone is moved on circular line in the plane PQO.

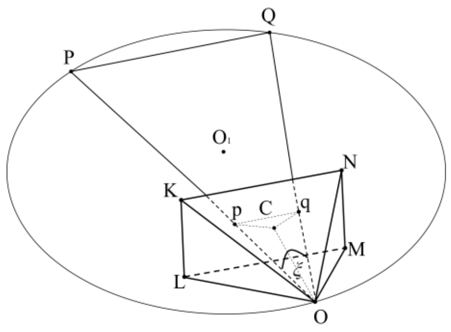


Figure 1. Trajectory of point O on the plane OPQ.

3.1. *Solve for the Angle ξ*

The horizontal field of view angle $\angle LOM$, the vertical field of view angle $\angle NOM$, and the physical focal length OC of the camera are obtained by calibrating the device camera, where OC is also the distance from the camera optical center O to the camera imaging plane $KLMN$.

The physical length of KL and KN can be obtained from the orthogonal equation:

$$\begin{aligned} KL &= 2 \cdot \tan(\theta_v/2) \cdot OC \\ KN &= 2 \cdot \tan(\theta_h/2) \cdot OC \end{aligned} \tag{1}$$

where θ_h is $\angle LOM$, and θ_v is $\angle NOM$.

Known image coordinates are projected on the imaging planes p and q; for the center image coordinates (C) of the imaging plane, the imaging plane height is represented by h_{img}, width is represented by w_{img}, and the physical length of the imaging plane is represented by KL and KN; the physical distance between p, q, and C is obtained by scaling t:

$$\begin{aligned} t &= \frac{\sqrt{KL^2+KN^2}}{\sqrt{{h_{img}}^2+{w_{img}}^2}} \\ Cp &= Cp_{img} \cdot t \\ Cq &= Cq_{img} \cdot t \\ pq &= pq_{img} \cdot t \end{aligned} \tag{2}$$

where Cp_{img}, Cq_{img}, and pq_{img} are the Euclidean distances between image coordinates, and Cp, Cq, and pq are the physical lengths in the real world. Combining the physical focal length OC and Pythagorean theorem, the physical length of Op and Oq can be found. According to the cosine law, we can solve for ξ as follows:

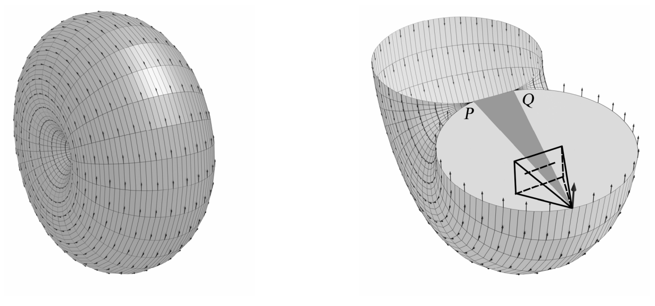
$$\xi = \arccos\left(\frac{Op^2 + Oq^2 - pq^2}{2 \cdot Op \cdot Oq}\right) \tag{3}$$

3.2. *Definition of Parametric Equation for Torus-like Surfaces*

According to Figure 1, knowing the world coordinates of points P and Q and their angle ξ at point O, the radius R of the circular line, with O_1 as the center and PQ as the chord, and the distance h from the center O_1 to the chord PQ can be found:

$$\begin{aligned} R &= \sqrt{\frac{PQ^2}{2\cdot(1-\cos(2\cdot\xi))}} \\ h &= R \cdot \cos(\xi) \end{aligned} \tag{4}$$

Figure 1 is just the trajectory of a point O on a plane, and there are countless such planes in world coordinate systems, each of which constructs a segment of arcs with PQ as the chord. These arcs will rotate with PQ as the axes to obtain a rotary body shaped like Figure 2a.



(**a**) (**b**)

Figure 2. Torus-like surface field. (**a**) Complete; (**b**) lower half.

To facilitate the observation of the position of the view cone in relation to this rotated body, the lower half of it is given in Figure 2b. We will refer to the surface of this rotated body as the torus-like surface (TLS), where all points on the TLS are the possible solution set of camera position O.

It is easy to see from the generation process that a TLS at any position in the world coordinate system is obtained by rotating an arc around any axis. Taking the PQ axis as an example, its parametric equations are essentially the parametric equations of a spatial circumference with radius h determining the trajectory of the center of the circular arc and determining the size of the arc based on the center of the spatial circumference with radius R.

For the general equation for the spatial circumcircle with the midpoint of the PQ axis as the center (x_{mid}, y_{mid}, z_{mid}), vector $\boldsymbol{PQ}$ as the normal vector (A_1, B_1, C_1), and h as the radius, Equation (5) represents the trajectory obtained by rotating the center of the circle O_1 around the $\boldsymbol{PQ}$ in Figure 1:

$$\begin{cases} \left(x_{O_1} - x_{mid}\right)^2 + \left(y_{O_1} - y_{mid}\right)^2 + \left(z_{O_1} - z_{mid}\right)^2 = h^2 \\ A_1\left(x_{O_1} - x_{mid}\right) + B_1\left(y_{O_1} - y_{mid}\right) + C_1\left(z_{O_1} - z_{mid}\right) = 0 \end{cases} \tag{5}$$

To facilitate the solution, it needs to be converted into a parametric equation. Let $X = x_{O1} - x_{\mathrm{mid}}$, $Y = y_{O1} - y_{\mathrm{mid}}$, and $Z = z_{O1} - z_{\mathrm{mid}}$ for Equation (5). Combining the two equations, the Z variables and formulas are deformed to obtain Equation (6):

$$\frac{\left({A_1}^2 + {B_1}^2 + {C_1}^2\right){C_1}^2}{{B_1}^2 + {C_1}^2}X^2 + \left(\frac{A_1 B_1}{\sqrt{{B_1}^2 + {C_1}^2}}X + \sqrt{{B_1}^2 + {C_1}^2}Y\right) = h^2 {C_1}^2 \tag{6}$$

To convert (6) into a more easily computable parametric equation, let

$$\begin{cases} X = h\sqrt{\frac{({B_1}^2 + {C_1}^2)}{({A_1}^2 + {B_1}^2 + {C_1}^2)}}\cos u \\ \frac{A_1 B_1}{\sqrt{({B_1}^2 + {C_1}^2)}}X + \sqrt{({B_1}^2 + {C_1}^2)}Y = aC_1 \sin u \end{cases} \tag{7}$$

Substituting (7) into (6) yields the parametric equation of Y with respect to u. Subsequently, substituting the parametric equations of X and Y with respect to u back into (5) yields the parametric equation of Z with respect to u. The parametric equations of X, Y, and Z are shown in Equation (8), where $0 \leq u \leq 2\pi$:

$$\begin{cases} X = h\sqrt{\frac{({B_1}^2 + {C_1}^2)}{({A_1}^2 + {B_1}^2 + {C_1}^2)}}\cos u \\ Y = \frac{h}{\sqrt{{B_1}^2 + {C_1}^2}}\left(C_1 \sin u - \frac{A_1 B_1}{\sqrt{{A_1}^2 + {B_1}^2 + {C_1}^2}}\cos u\right) \\ Z = -h\left(\frac{B_1}{\sqrt{{B_1}^2 + {C_1}^2}}\sin u + \frac{A_1 C_1}{\sqrt{{A_1}^2 + {B_1}^2 + {C_1}^2}\sqrt{{B_1}^2 + {C_1}^2}}\cos u\right) \end{cases} \tag{8}$$

Ultimately, the parametric equation for O_1, as in Figure 1, can be obtained as follows:

$$\begin{cases} x_{O_1} = x_{mid} + X \\ y_{O_1} = y_{mid} + Y \\ z_{O_1} = z_{mid} + Z \end{cases} \tag{9}$$

When both B_1 and C_1 are 0, Equation (9) will fail, and the spatial circumferential parametric equation is converted to the parametric equation of a circular line on the $x = x_{\mathrm{mid}}$ plane:

$$\begin{cases} x_{o_1} = x_{mid} \\ y_{o_1} = y_{mid} + h\cos u \\ z_{o_1} = z_{mid} + h\sin u \end{cases} \tag{10}$$

The above parametric equation represents the trajectory of the outer circle center O_1 rotating around the axis PQ.

Then, take the point O_1 as the center of the circular line, R as the radius, and cross-multiply the vector $\boldsymbol{PO}_1$ with the vector $\boldsymbol{QO}_1$ to obtain the normal vector (A_2, B_2, C_2) to construct the parametric equations of the spatial circumference of the arc, which is also divided into two cases, $0 \leq v \leq 2\pi$, when both B_2 and C_2 are not 0:

$$\begin{cases} x_o = x_{o_1} + R\sqrt{\frac{({B_2}^2+{C_2}^2)}{({A_2}^2+{B_2}^2+{C_2}^2)}}\cos v \\ y_o = y_{o1} + \frac{R}{\sqrt{{B_2}^2+{C_2}^2}}\left(C_2\sin v - \frac{A_2B_2}{\sqrt{{A_2}^2+{B_2}^2+{C_2}^2}}\cos v\right) \\ z_o = z_{o1} + -R\left(\frac{B_2}{\sqrt{{B_2}^2+{C_2}^2}}\sin v + \frac{A_2C_2}{\sqrt{{A_2}^2+{B_2}^2+{C_2}^2}\sqrt{{B_2}^2+{C_2}^2}}\cos v\right) \end{cases} \tag{11}$$

When both B_2 and C_2 are 0,

$$\begin{cases} x_o = x_{o_1} \\ y_o = y_{o_1} + r\cos v \\ z_o = z_{o_1} + r\sin v \end{cases} \tag{12}$$

Equations (9) and (10), along with Equations (11) and (12), are the four forms of the parametric equations for TLS obtained according to different combinations of each other.

3.3. Solving the Intersection of Three Torus-like Surfaces

Most of the methods for intersecting two curves use tracking method, where the definition of the step size is difficult after the initial position of the intersection point is determined. In order to ensure accurate intersection, the traditional approach is to repeatedly experiment with the optimal step size to adapt to the current surface intersection. However, the sizes of torus-like surfaces change dynamically in application scenarios, so it is obvious that a fixed step size cannot be well adapted to torus-like surface intersection. Different from the traditional methods, this paper adopts a method to intersect torus-like surfaces with an adaptive change of step size according to the characteristics of the torus-like surfaces, which relates the step size to the radius of curvature at the end of the intersecting line, so that the step size can be changed in real time according to the real characteristics of the surfaces. The points on the surface intersection line are fitted with five non-uniform B-spline curves to obtain a higher accuracy of localization.

Four PCs allow the construction of parametric equations for three TLSs at three different locations.

The intersection between two of the parametric TLSs is transformed into a set of parametric curves and a parametric TLS, i.e., the problem is transformed into a problem of intersection between a parametric curve and a parametric surface. Let the two TLSs be $P_1 = P_1(u_1, v_1)$, $P_2 = P_2(u_2, v_2)$. Define a u-directed parametric curve on P_1 as $C(u)$, whose equation can be expressed as $C(u) = P_1(u_1, \overline{v_1})$, where the parameter $\overline{v_1}$ indicates that the parameter is fixed. $c_0 = P_1(u_{10}, \overline{v_1})$ is a point on the parametric line $C(u)$ of the torus-like surface P_1, and $s_0 = P_2(u_{20}, v_{20})$ is a point on the torus-like surface P_2. The vector distance between c_0 and s_0 is defined as $\tau = c_0 - s_0$, and an accuracy threshold ε is set; when $|\tau| \leq \varepsilon$, it means that the distance between the two points is small enough to be used as the initial value of the intersection. The variable in the parametric line $C(u) = P_1(u_1, \overline{v_1})$ is u1, and the corresponding variables of surface P_2 are u_2 and v_2, respectively. Let

$$\begin{cases} \Delta u_1 = u_1 - u_{10} \\ \Delta u_2 = u_2 - u_{20} \\ \Delta v_2 = v_2 - v_{20} \end{cases} \tag{13}$$

A Taylor's formula expansion of τ is obtained by neglecting the higher terms:

$$\begin{cases} \tau \approx P_{1u}(u_{10}, \overline{v}_1)\Delta u_1 - P_{2u}(u_{20}, v_{20})\Delta u_2 - \\ \qquad P_{2v}(u_{20}, v_{20})\Delta v_2 \end{cases} \tag{14}$$

Projecting the above equation into normed linear space and dot-multiplying P_{1u}, P_{2u}, and P_{2v} on each side of the equation yields three linear equations:

$$\begin{cases} P_{1u} \cdot \tau = (P_{1u} \cdot P_{1u})\Delta u_1 - (P_{1u} \cdot P_{2u})\Delta u_2 - (P_{1u} \cdot P_{2v})\Delta v_2 \\ P_{2u} \cdot \tau = (P_{2u} \cdot P_{1u})\Delta u_1 - (P_{2u} \cdot P_{2u})\Delta u_2 - (P_{2u} \cdot P_{2v})\Delta v_2 \\ P_{2v} \cdot \tau = (P_{2v} \cdot P_{1u})\Delta u_1 - (P_{2v} \cdot P_{2u})\Delta u_2 - (P_{2v} \cdot P_{2v})\Delta v_2 \end{cases} \tag{15}$$

The above system of three equations in groups does not have a zero denominator or does not converge because the coefficient matrix is positively definite. Substituting u_{10} + Δu_1, u_{20} + Δu_2, v_{20} + Δv_2 into τ by using Taylor's formula expansion for the point c_0 and iterating based on the normed linear space projection method [25] until $|\tau| \leq \varepsilon$, the initial value of the intersection point is obtained, and whether the intersection point is on the third TLS or not is determined, and if it is, it is saved as a row of intersection candidates.

The next intersection point is solved next, and so on; a series of intersections can be obtained to acquire the whole intersection line. Requirements to take the next intersection are necessary to analyze the point in the space curve of the characteristic parameters. c_0 = $c(s_0)$ is the curve c on a point, s_0 is the arc length parameter, and then c_0 neighboring points can have a Taylor expansion expressed as follows:

$$c(s_0 + \Delta s) = \dot{c}(s_0) + \dot{c}(s_0)\Delta s + \frac{1}{2'}\ddot{c}(s_0)\Delta s^2 + o(\Delta s^2) \tag{16}$$

There are in differential geometry:

$$\dot{c}(s_0) = \alpha; \ddot{c}(s_0) = k\beta \tag{17}$$

In the [c_0, α_0, β_0] coordinate system, defining the point at c_0 as the starting point, there is an arc length parameter of s_0 = 0, step size of Δs = s, and the neighboring points can be expressed as follows:

$$c(s) = c_0 + \alpha_0 s + \frac{1}{2}ks^2\beta_0 \tag{18}$$

where α, β, and k are the tangent vectors, principal normal vectors, and curvature of the space curve c at the point c_0. Based on α and β, the close planes can be obtained, which in turn gives the geometric structure of the point c_0 on the space curve [26]; we can solve for α, β, and k, respectively:

$$\begin{array}{l} \cos\theta = \boldsymbol{n}_1 \cdot \boldsymbol{n}_2 \\ \dot{u} = \frac{du}{ds}; \dot{v} = \frac{dv}{ds} \\ k_n = \frac{II}{I} = L(\dot{u})^2 + 2M\dot{v} + N(\dot{v})^2 \\ \alpha = \frac{n_1 \times n_2}{|n_1 \times n_2|} \\ k = |\boldsymbol{k}| = \frac{\sqrt{(k_n^1)^2 + (k_n^2)^2 - 2k_n^1 k_n^2 \cos\theta}}{|\sin\theta|} \\ \beta = \boldsymbol{k}/k \end{array} \tag{19}$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are the normal vectors at the point c_0; k_n is the normal curvature of each surface at that point obtained from the first and second fundamental forms of the surface; L, M, and N are the coefficients in the second fundamental form; and $\boldsymbol{k}$ is the curvature vector.

The intersection, which is solved by plugging α, β, and k combined with s into Equation (18), is an approximation, and at this time, the step size s is unknown, and if it is set artificially based on experience, the result is not stable and efficient enough, and we found that TLS intersection lines have more obvious curvature changes in different regions. So, we correlate the step-size s with the radius of curvature at the end of the intersection line according to [27] so that the step size can be changed in real time based on the real characteristics of the surface; points on the intersection line are fitted using fivefold non-uniform B-spline curve.

$$\left\{\begin{array}{l} k_r = |C''(u)|/(1+C'(u)^2)^{3/2} \\ r = 1/k_r \\ s = 2 \times \sin(\theta/2) \times r; \theta \leq 2 \times \arccos(1 - e \times k_r) \end{array}\right. \tag{20}$$

where C(u) is the fivefold non-uniform B-spline curve equation, u = 1 when the endpoint of C(u), r is the radius of curvature on C(u), and k_r is the curvature of C(u). When k_r is less than 0, it indicates that the fitted curve curves are bent clockwise along the direction of advancement, and when it is greater than 0, it indicates that it is bent counterclockwise. e denotes the threshold of accuracy, which is generally taken to be 0.001 mm. The points obtained at this time are not the exact solution, and it is also necessary to iterate to obtain a more accurate value [28].

In summary, two points closer to the point $c(s)$, denoted as p_1 and p_2, are firstly selected on the two TLSs P1 and P2, respectively, with p_1 belonging to P1 and p_2 belonging to P2.

If there is a point p_1 and p_2 converging to the neighboring point $c(s)$, the point should meet the set accuracy $|p_1 - p_2| \leq \varepsilon$. If p_1 and p_2 cannot converge at the neighboring point, p_2 is defined as an approximation of the intersection point. Then, the above computation process is repeated continuously.

If the condition is satisfied, its coordinates need to be substituted into the equation of the third TLS in order to determine whether the point is on this TLS or not. If the distance d from the midpoint of the axis to the intersection point satisfies the following condition, then γ is the angle between the vector from the midpoint of the axis to the intersection and the axis vector.

$$\tau \geq \left\{\begin{array}{l} |d - (h+R)|; \gamma \approx \frac{\pi}{2} \\ \left|d - (\sqrt{R^2 - h^2}/\cos\gamma)\right|; 0 < \gamma < \frac{\pi}{2} \end{array}\right. \tag{21}$$

The point is added to the candidate set of intersections. Then, the exact point $c(s)$ found by iteration is used as the initial intersection, and the location of the next intersection continues to be counted until the boundary of the surface is traced.

The approximate process for solving the intersection is as Figure 3:

The three TLS equations are constructed in such a way as to obtain the circumferential angles ξ_1, ξ_2, and ξ_3, respectively. The angle of circumference $\xi_{1'}$, $\xi_{2'}$, and $\xi_{3'}$ formed by the resulting point of intersection and the two endpoints on the axis, if it satisfies $\xi_{1'} > \xi_1$ or $\xi_{2'} > \xi_2$ or $\xi_{3'} > \xi_3$, shows that the intersection is on the minor arc opposite the axis and should be excluded.

There may be more than one candidate for the intersection, which needs to be filtered according to the normal vectors of the three TLSs' tangent at the intersection. If the intersection meets the requirements, then it must meet the three TLSs of the visual cone on this intersection overlap, i.e., the upper vector of the three must be at the intersection of the same tangent. Thus, it can be based on the intersection of the tangent surface of the normal vector to determine whether it is colinear or not. If not, then rule out the intersection. In practice, there will still be a very small error, so it is necessary to set a threshold, the normal vector between the two cross-multiplications of the modulus, to do the error evaluation and set it to 1×10^{-11} for the best results after many tests. After filtering, there may still be more than one intersection, so direction vectors should be utilized for further filtering.

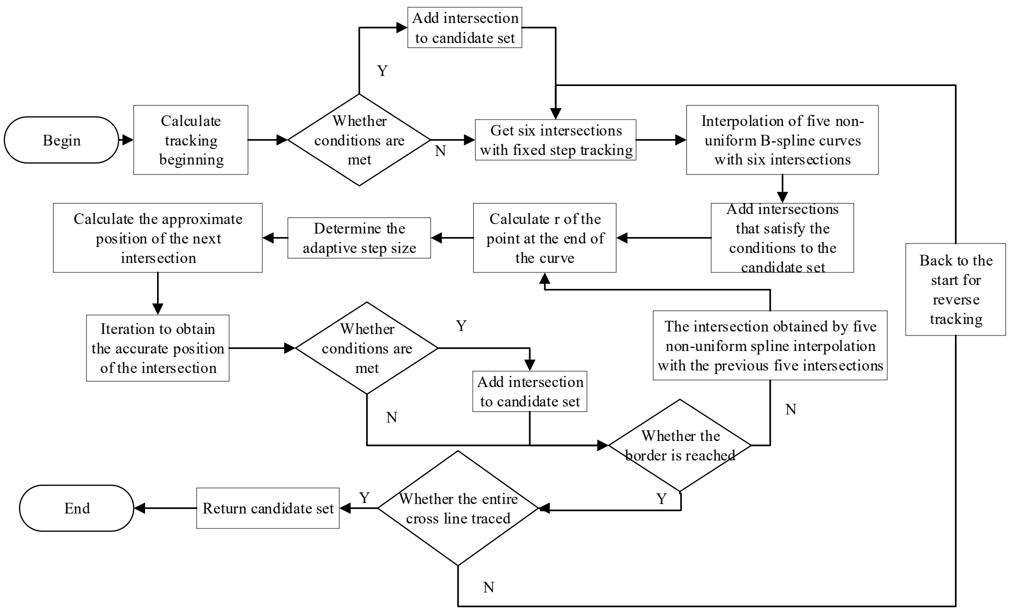

Add intersection to candidate set
Y
Begin
Calculate tracking beginning
Whether conditions are met
N
Get six intersections with fixed step tracking
Interpolation of five non-uniform B-spline curves with six intersections
Calculate the approximate position of the next intersection
Determine the adaptive step size
Calculate r of the point at the end of the curve
Add intersections that satisfy the conditions to the candidate set
Back to the start for reverse tracking
Iteration to obtain the accurate position of the intersection
Whether conditions are met
Y
Add intersection to candidate set
N
The intersection obtained by five non-uniform spline interpolation with the previous five intersections
N
Whether the border is reached
Y
End
Return candidate set
Y
Whether the entire cross line traced
N

Figure 3. Process for obtaining the candidate set of intersections.

3.4. Solving Forward Vector and Upper Vector

The world coordinates of the intersection O are shown in Section 3.3, and we can easily obtain the distances of OP and OQ. Find the world coordinates S_0 and S_1 of P, Q projected on the imaging plane by the proportionality of OP and OQ to Op and Oq derived in Section 3.1:

$$\begin{aligned} S_0(x,y,z) &= (O(x,y,z) - P(x,y,z)) \cdot (\tfrac{Op}{OP}) + O(x,y,z) \\ S_1(x,y,z) &= (O(x,y,z) - Q(x,y,z)) \cdot (\tfrac{Oq}{OQ}) + O(x,y,z) \end{aligned} \tag{22}$$

Follow the above equation to find the world coordinates of third point projected on the imaging plane, assumed to be S_2.

The forward vector $\boldsymbol{f}$ is obtained by vector cross-product:

$$\boldsymbol{f} = \boldsymbol{S_0S_1} \times \boldsymbol{S_1S_2} \tag{23}$$

Since these four feature points are ordered beforehand on the image coordinate system according to their positions from top to bottom and left to right, it can be guaranteed that the forward vector is the orientation of the view cone. At the same time, it can be guaranteed that the upper vector of the subsequent solution always points to the camera directly above. If there are still multiple intersection solutions, use whether the angle between $\boldsymbol{OP}$ and $\boldsymbol{f}$ is greater than half of the diagonal viewing angle of the view cone. If it is greater, it means the point is a mirror solution; exclude this type of intersection and leave the intersection with the smallest error as the ultimate solution.

To solve the upper vector based on the front vector, it is also necessary to find a point T on the centerline JI on the imaging plane and parallel to the KN side that is different from the pixel position of point C. As shown in Figure 4, select the image coordinate point p and determine the image coordinates of point T, whose horizontal coordinate is the abscissa of p, and the vertical coordinate is the ordinate of C. In this respect, a right triangle pTC can be constructed, and β is found based on the Pythagorean theorem, and then the vector $\boldsymbol{OT}$ can be obtained from the vector $\boldsymbol{OS_0}$ rotated around the forward vector $\boldsymbol{f}$ by an angle β. The upper vector $\boldsymbol{u}$ of the camera position point O can be obtained by cross-multiplying the unit vector of $\boldsymbol{OT}$ with the forward vector $\boldsymbol{f}$.

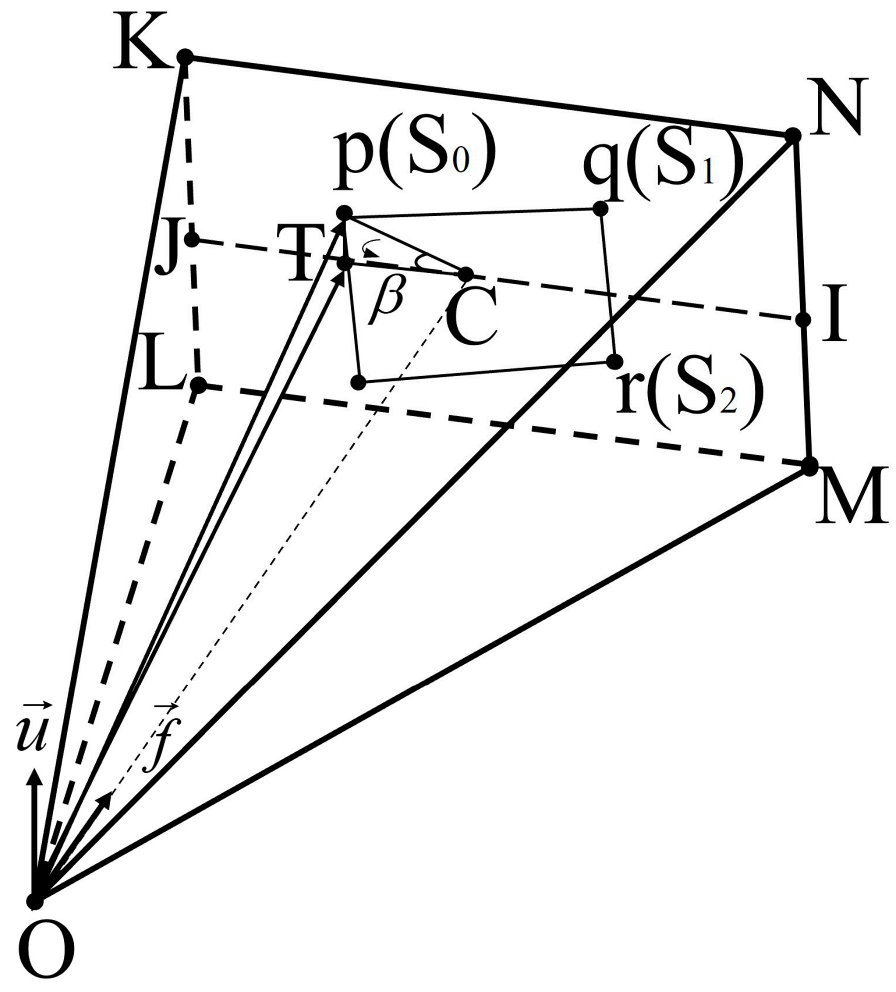


Figure 4. Solving the upper vector based on the forward vector.

The TLS solution yields the forward and upward vectors of the camera in the world coordinate system, which can be extrapolated by Rodriguez's formula to obtain the rotation matrix $\boldsymbol{R}$. Combined with the intersection coordinate O, the translation vector $\boldsymbol{t}$ is easily obtained.

4. Experiments and Analysis of Results

Given the true query camera center $O_g = -\boldsymbol{R}_g{}^T\boldsymbol{t}_g$ and the estimated query camera center O, the position error is calculated as $||O - O_g||$. The rotation error is calculated as $||\ln(\boldsymbol{R}_g\boldsymbol{R}^T)||$. While arccos((trace($\boldsymbol{R}$) − 1)/2) is often used as the rotation error metric, using $||\ln(\boldsymbol{R}_g\boldsymbol{R}^T)||$ is more stable for small angle rotations.

To support the UP2P algorithm application, the method in [29] is utilized to decompose the ground truth rotation matrix $\boldsymbol{R}_g$ into $\boldsymbol{R}_Y$ rotated around the Y-axis and $\boldsymbol{R}_{XZ}$ rotated around the X-Z plane vectors, i.e., $\boldsymbol{R}_g = \boldsymbol{R}_Y\boldsymbol{R}_{XZ}$. $\boldsymbol{R}_{XZ}$ is taken as the input, and the only rotation unknown θ is the rotation around the Y-axis.

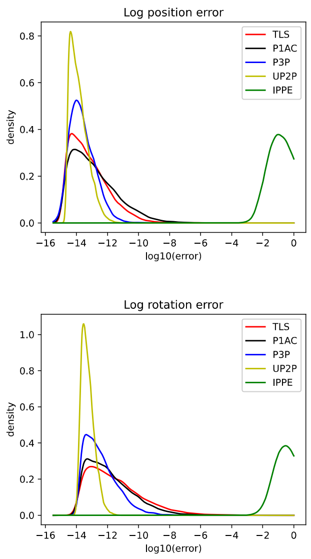
To support the IPPE algorithm application, three additional PCs are generated by sampling 2D points around point x in the reference image using the method in the P1AC literature. Then, 3D points are obtained by intersecting a ray with the plane where the normal vector $\boldsymbol{n}$ is located through the camera as a starting point. Additionally, 2D points are found by applying an affine transformation to simulate four 3D points coplanar; three of these four points are not co-linear with each other and will be used as inputs to the method based on four PCs.

4.1. Synthetic Data Experiments

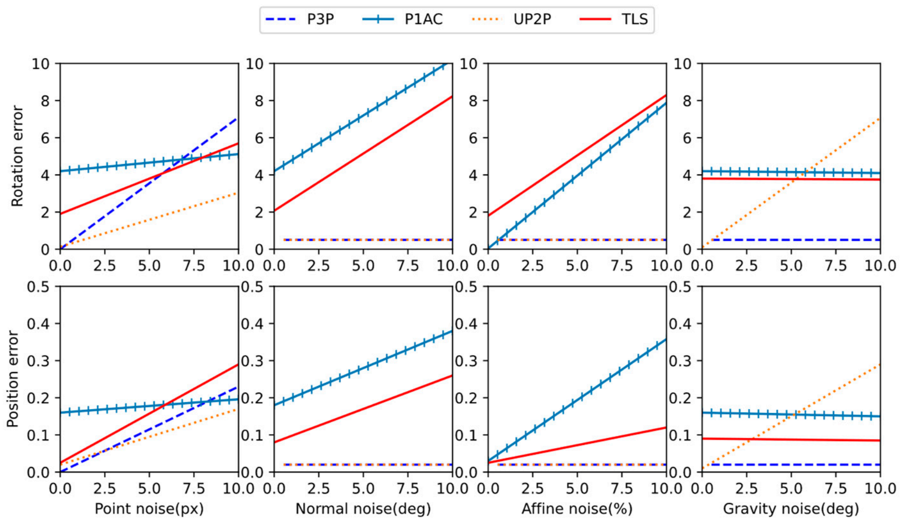
We use a data generation method adapted from [30], assuming that reference and query cameras with the same intrinsic are randomly placed at a distance of [1,2] around the origin and that both cameras are oriented towards a random target point in the range of $[-0.5,0.5]^3$. To establish a single-point correspondence between the two cameras, we randomly select a 3D point from N (0, $I_{3\times3}$), then take a random plane normal vector at that point and project it onto the imaging surface of the two cameras using a focal length of 400. Then, the affine transformation between two cameras is computed from the local homography matrix using the method of reference [31] by transforming the two cameras and the 3D point so that the reference camera's external reference matrix with respect to the origin is $[\mathbf{I}|\mathbf{0}]$. The translation vectors and target 3D points in the synthesized data are generated by the rand() function of C++, and these random numbers obey a normal distribution; the rotation matrix between the reference and query cameras is transformed by a random unit quaternion that follows a uniform distribution law on SO(3).

4.1.1. Numerical Stability

We tested the numerical stability of each solver on 10,000 random problem instances with no noise added to the observations. Figure 5a shows the density plots of log rotation and position errors for each solver. Except for IPPE, which performs poorly due to the unavoidable errors in the three virtual PCs generated by the P1AC literature's method. IPPE is quite sensitive to PCs, whether it is completely coplanar. On the contrary, the TLS method is not much affected. It is known that TLS adapts itself according to the point positions and solves stably even at the four points of the near coplanar plane without any peaks higher than 10^{-2}. The fact that more than 99.9% of the points in P1AC and TLS have an error of less than 10^{-5} suggests that these solvers are able to produce usable results.



(**a**)



(**b**)

Figure 5. Results of synthetic data experiments. (**a**) Analysis of numerical stability with zero noise added to observations. The plots are estimates of the distribution produced by Gaussian kernel-density estimation. Top: Log rotation error. Bottom: Log position error. (**b**) The median error of two-dimensional observations, normal vectors, affine transformations, and gravity vectors relative to noise. The x-axis shows the std. dev. of the noise added.

4.1.2. Noise

We test the sensitivity of various methods by adding multiple types of noise to the generated data.

- Point noise: add Gaussian noise to 2D observations;
- Normal noise: randomly select angles in the Gaussian distribution, and the normal vector is rotated according to the angles;
- Affine noise: adds Gaussian noise to the elements of the 2×2 affine transform, and the level of Affine noise is chosen according to the selected percentage error, e.g., with a percentage of r, the noise added to $\mathrm{a_{ij}}$ is $\mathrm{r{\cdot}a_{ij}}$;
- Gravity noise: perform a random rotation of $\boldsymbol{R}_{XZ}$, with the rotation angle taken from a normal distribution.

The TLS, P1AC, UP2P, and P3P solvers were tested in a range of noise settings for each noise type, and the median error was calculated for 10,000 iterations at each setting. For the noise types, which varied in each setting, the default noise settings were 1-pixel noise, 1° normal vector noise, 4% affine noise, and 0.5° gravity-oriented noise.

From the results in Figure 5b, it can be seen that since the solution process of P1AC relies heavily on normal vectors and affine matrices, it is extremely sensitive to normal vector noise and affine noise, but since the algorithm only uses a single point correspondence, it is less affected by point correspondence; UP2P introduces gravity vectors as an input, which is not only affected by point noise, but also quite sensitive to gravity vector noise; P3P does not P3P does not utilize normal vectors and affine matrices in the solution process, nor does it involve gravity vectors, but it is more affected by the feature points; similarly, the model of TLS is constructed based on geometric constraints between 3D points and their projected points, and thus it is largely affected by the point correspondences.

Theoretically, TLS is not affected by normal and affine noise. In order to compare with P1AC in normal noise and affine noise experiments, the three virtual points inputted into TLS are generated according to the method in P1AC. This process utilizes the normal vector and affine matrix, which is equivalent to indirectly affecting the point correspondence through normal noise and affine noise. The results show that as the normal noise and affine noise increase, the magnitude of error change is less affected by these noises than P1AC, i.e., the TLS can solve the high-accuracy solution more stably.

The accuracy of TLS is weaker than P1AC or even P3P with the addition of roughly 5px point noise to the default observations, which is not too much of a concern in practice, where the RANSAC algorithm is generally used to exclude feature pairs with high point noise during feature matching, and will be previewed in the subsequent real data experiments. Subsequent real data experiments will pre-process the matched pairs using RANSAC.

4.1.3. Time

We compared the average time required by each solver to solve a single configuration problem out of 10,000 random problems. All solvers were implemented in C++, computed using an Intel(R) Core(TM) i7-7700 (Intel Products (Chengdu) Co., Ltd., Chengdu, China), and used the UP2P and P3P solvers from the PoseLib library, with the execution of IPPE provided by the solvePnP function in OpenCV and P1AC using the author's official release. The results are shown in Table 1. With the exception of IPPE, the average time for the rest of the solvers is less than 2.8 µs, and solving is quite fast.

Table 1. Average timing in µs over 10,000 trials.

	P3P	**UP2P**	**IPPE**	**P1AC**	**TLS**
Time (µs)	0.54	0.33	25.27	2.73	2.37

4.1.4. Robust Estimation

We investigated the performance of TLS and P3P in robust estimation with an increasing proportion of outliers.

To analyze the trade-off between having a small minimum sample size and a large noise sensitivity, we tested a range of outlier ratios from 0 to 0.9, with 10,000 trials at each setting, with the default settings for noise consistent with those in the noise experiments. In each trial, we randomly generated 1000 PCs, replacing some of the correspondences with random values based on the desired outlier ratio. We then run vanilla RANSAC or LO-RANSAC [32] using TLS or P3P as the minimum solver in local optimization. LO-RANSAC reduces the effect of noise on the minimum solver by combining iterations with local optimization (LO) to increase the set of interior points from the initial estimate. The effect of noise on the minimum solver is minimized.

The average error of each method is stable over a range of outlier ratios. Table 2 shows the average error for both methods for all outlier ratios. When vanilla RANSAC is used, TLS is less accurate than P3P at this noise level, and when LO-RANSAC is used, although TLS has a slightly higher error than P3P, both methods have very low error ratios, proving their effectiveness. Note that on real data, TLS performs substantially better than P3P.

Table 2. Average rotation and pose error for each robust estimation method on synthetic data over a range of outlier ratios.

Robust Estimation	**TLS**		**P3P**	
	Rot (°)	**Pos.**	**Rot (°)**	**Pos.**
RANSAC	0.462	0.010	0.057	0.002
LO-RANSAC	0.079	0.002	0.017	0.001

Figure 6 shows the average time for each method. With vanilla RANSAC, TLS is faster than P3P at outlier ratios of about 0.37; with LO-RANSAC, TLS is faster than P3P at all outlier ratios. This result shows that our TLS solver is accurate enough to provide a good initialization for LO-RANSAC and reaches convergence faster than P3P, even at lower outlier ratios. In the real data experiments (Section 4.2), in order to be consistent with the state-of-the-art, we used Graph Cut RANSAC (GC-RANSAC) [6], which improves on LO-RANSAC by adding graph cut optimization to the LO iterations. We did not use GC-RANSAC in the synthetic data experiments because the observations are randomly generated and not spatially coherent.

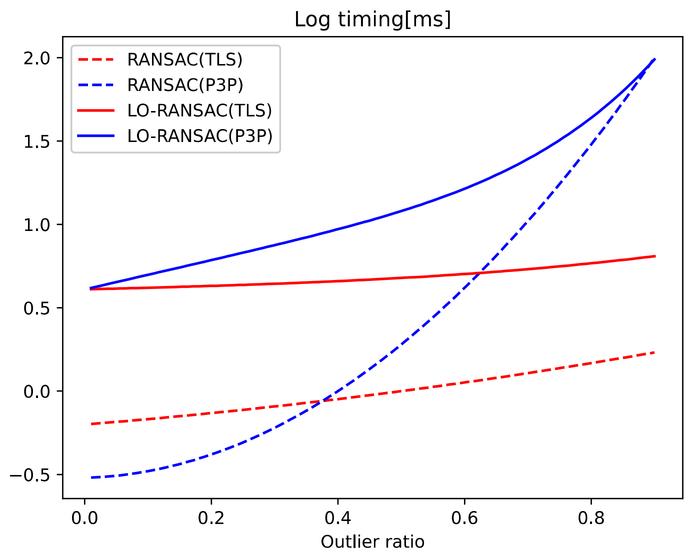


Figure 6. Average timing for each robust estimation method across a range of outlier ratios.

Mirror solutions, i.e., pairs of solutions that are rotated by exactly 180 degrees from each other, may occur in the TLS solution. Although these solutions will be filtered out by Ransac, we have added an extra restriction to the algorithm, i.e., if the angle between a ray vector emitted from the position out and the angle formed by the forward vectors is obtuse, the solution can be recognized as a mirror solution, and another solution is taken.

4.2. Real Data Experiments

To compare the performance of the methods on large-scale image localization datasets, we use the Cambridge Landmarks [33] and Aachen Day-Night v1.1 [34] datasets. These two datasets are more commonly used in the image-based visual localization literature.

The Cambridge Landmarks dataset consists of six scenes from Cambridge, UK, each recorded from multiple video sequences taken on a smartphone, capturing different parts of the city. From the sequences recorded for each scene, one part represents the database image for that scene, and the other part is used to acquire query images. The ground truth and internal intrinsic parameter calibrations of all images were determined using Visual SFM [35] Visual (SfM) software (v0.5.26). The reference and query images are taken from the same walking paths under the same lighting and similar environmental conditions, which increases the difficulty of localization due to the fact that the angles and shooting distances are not sampled from similar trajectories. Localization performance was generally determined by comparing the median errors of position and orientation. In addition, we evaluated the proportion of images (i.e., recall) for which the positional estimates were within 5 cm/1°, 10 cm/1°, and 20 cm/1° of the actual position.

The Cambridge Landmarks dataset was taken under the same lighting and similar environmental conditions. We choose the dataset Aachen Day-Night v1.1 to compare the localization performance of each algorithm based on a daytime reference image for the daytime query Day and a nighttime query Night, which implies that there is no nighttime image in the reference image. These images were taken at different times and years over a two-year period. Therefore, these images cover a wide range of lighting and weather conditions, as well as temporary occlusion of construction sites and building changes that are not present in the 3D model. The ground truth of the daytime reference images was established via COLMAP [36], and the nighttime query images were re-referenced to the initial pose for alignment [34]. We followed a common evaluation protocol to report the percentage of images localized within three error thresholds (0.25 m/2°, 0.5 m/5°, 5.0 m/10°).

There are various ways to obtain affine features from real images. Experiments will use the following two feature extraction and matching algorithm approaches to obtain valid 2D feature point pairs as well as affine matrices. One is to use a local feature detector (e.g., DoG [37]) to estimate the location and scale of key points and then use patch-based AffNet [38] to obtain the affine shapes. Finally, a patch-based descriptor such as Hard-Net [39] is applied. The other uses SuperPoint [40] for feature detection, LightGlue [41] for feature matching, and finally SelfScaleOri [42] to obtain the image feature orientation and scale. Based on the orientation and scale, the affine matrix A_i of the i-th feature is approximated as $S_{qi}R_{\beta i}$, where S_{qi} is a diagonal matrix scaled by the detected scale factor q_i scales uniformly along the axis, and $R_{\beta i}$ belonging to SO(2) is rotated by the estimated angle $\beta_i \in [0, 2\pi)$. The normal vector is estimated using the 200 nearest neighbors of each point in the SfM point cloud. The direction of gravity is extracted from the ground truth of the rotation matrix as UP2P input.

In these experiments, our aim is to analyze the usefulness of TLS in practice. For this purpose, we use GC-RANSAC and fine-tune each solver so that they can achieve the best performance possible.

As can be seen from Tables 3 and 4, the P1AC method has a smaller average error and a higher percentage of recall in almost all scenarios. We also find that TLS has comparable results with P1AC in terms of positional error, but the difference is larger in terms of rotational error. According to the TLS principle, the higher the positional solving accuracy, the smaller the rotational error will be. Next, we attempt to utilize SuperPoint + LightGlue + SelfScaleOri (SP + LG + SSO) to obtain the image feature matching, feature orientation, and scale to test performance.

Table 3. Cambridge Landmarks median position (centimeters) and rotation (degrees) errors of GC-RANSAC combined with various solvers when using DoG + AffiNet + HardNet.

	Position (cm) ↓					**Rotation (°) ↓**				
Scene	**P3P**	**P1AC**	**UP2P**	**IPPE**	**TLS**	**P3P**	**P1AC**	**UP2P**	**IPPE**	**TLS**
Great Court	8	**6**	**6**	7	**6**	0.1	0.09	**0.08**	0.1	0.12
King's Col.	11	**7**	8	10	8	0.16	**0.14**	0.15	0.21	0.20
Old Hospital	9	7	7	8	**5**	0.14	**0.12**	0.13	0.13	0.15
Shop Façade	5	**2**	3	4	3	0.13	0.13	0.14	**0.12**	0.14
St Mary's Ch.	7	**5**	**5**	6	6	0.2	**0.17**	**0.17**	0.19	0.19
Street	30	22	43	221	**20**	0.58	0.36	0.79	**0.26**	0.42
Avg.	11	8	12	43	**8**	0.22	**0.17**	0.24	**0.17**	0.24
Weighted avg.	22	**14**	32	152	15	0.41	0.26	0.52	**0.21**	0.36

Table 4. Cambridge Landmarks recalls (in percentages), at 0.1 m/1° and 0.2 m/1° of GC-RANSAC combined with various solvers when using DoG + AffiNet + HardNet.

	Recall (0.1 m/1°) ↑					**Recall (0.2 m/1°) ↑**				
Scene	**P3P**	**P1AC**	**UP2P**	**IPPE**	**TLS**	**P3P**	**P1AC**	**UP2P**	**IPPE**	**TLS**
Great Court	61.7	73.5	73.8	65.3	**73.9**	86.4	**91.1**	90.1	86.2	90.3
King's Col.	49.3	64.6	65.1	57.7	**65.8**	79.2	**86.9**	84.7	85.4	84.9
Old Hospital	58.7	**72.3**	68.4	62.5	69.1	79.8	**93.5**	89.3	85.3	89.5
Shop Façade	85.3	**87.6**	86.3	84.3	85.8	94.4	**95.1**	93.1	92.3	92.7
St Mary's Ch.	61.6	**72.5**	70.1	67.5	72.1	80.7	**87.6**	85.2	84.3	85.7
Street	10.6	**19.7**	16.5	19.1	17.2	17.8	**29.6**	23.9	26.7	24
Avg.	54.5	**65.03**	63.37	59.4	63.98	73.05	**80.63**	77.72	76.7	77.85
Weighted avg.	30.45	**40.61**	38.21	37.39	38.54	43.89	**53.52**	49.34	50.1	49.69

As shown in Table 5, all the solvers have improved greatly in extracting feature information using SP + LG + SSO, especially the TLS method in this paper, which has more significant improvement in position and rotation due to more and more accurate 2D-3D valid matches obtained. It is worth noting that the rotation error of P1AC is instead larger than before. This is most likely because the derived affine relation approximation from the orientation and scale obtained by SSO is more biased than the one obtained by AffiNet.

Aachen Day-Night is a feature extracted using both DoG + AffiNet + HardNet (DG + AN + HN) and SP + LG + SSO. Since we do not know the direction of gravity, it is assumed to point directly downward (i.e., $[0, -1, 0]^T$). This is usually a safe assumption since people tend to take photos upright. The results are shown in Tables 6 and 7.

Table 5. Cambridge Landmarks average median position (centimeters) and rotation (degrees) errors and average recalls (in percentages) at 0.1 m/1° and 0.2 m/1° of GC-RANSAC combined with various solvers when using SP + LG + SSO.

Methods	**Position (cm)/Rotation (°) ↓**	**Recall (0.1 m/1°)/Recall (0.2 m/1°) ↑**
P3P	8/0.16	72.9/76.1
P1AC	7/0.23	85.7/88.6
UP2P	7/0.19	86.4/90.5
IPPE	13/**0.12**	84.2/88.4
TLS	**6**/0.13	**87.3/90.8**

Table 6. Aachen Day-Night pose error recalls, in percentages, at 0.25 m/2°, 0.5 m/5°, and 5.0 m/10° of GC-RANSAC combined with various solvers using DG + AN + HN.

	Recall (0.25 m/2°) ↑				
	P3P	P1AC	UP2P	IPPE	TLS
Day	61.9	62.2	62.1	62.0	**62.3**
Night	47.3	**51.4**	48.1	46.2	47.9
	Recall (0.5 m/5°) ↑				
Day	83.4	**84.6**	83.8	83.5	**84.6**
Night	60.3	**65.9**	64.7	63.8	62.7
	Recall(5 m/10°) ↑				
Day	96.0	95.9	95.7	96.0	**96.1**
Night	74.2	**82.3**	79.6	76.4	80.1

Table 7. Aachen Day-Night pose error recalls, in percentages, at 0.25 m/2○, 0.5 m/5○, and 5.0 m/10○ of GC-RANSAC combined with various solvers using SP + LG + SSO.

	Recall (0.25 m/2°) ↑				
	P3P	P1AC	UP2P	IPPE	TLS
Day	63.4	64.3	65.8	66.1	**66.4**
Night	61.3	69.7	67.5	65.8	**70.1**
	Recall (0.5 m/5°) ↑				
Day	79.6	83.3	86.7	87.1	**87.9**
Night	77.5	**87.2**	86.4	86.6	87.1
	Recall(5 m/10°) ↑				
Day	92.4	**97.3**	96.8	97.1	**97.3**
Night	92.8	98.9	97.9	98.3	**99.0**

Observation of Table 6 shows that most of the solvers achieve good results in the Day dataset but have uneven results at Night, indicating that DoG is not stable and efficient enough to capture valid 2D-3D matches at night.

Since the application of SP + LG + SSO in the original Day dataset enabled the various methods to achieve close to 100% recall, we resized the images so that their longest dimension is 800 pixels at maximum. On the Day sequence, all algorithms had reasonably good precision, except for P3P, which was slightly weaker. Overall, TLS performs best. Although P1AC achieves high recall on 5 m/10°, it is not accurate enough at tighter thresholds. In the Night sequence, TLS achieves higher accuracy with SP + LG + SSO feature extraction.

We used DG + AN + HN for feature extraction and GC-RANSAC for filtering to report the average time of various methods. As we expected, P3P has the shortest time in the Cambridge Landmarks dataset, P1AC has the shortest time for the Day sequences in Aachen Day-Night, and IPPE is faster for the Night sequences. TLS does not have much of an advantage in terms of time due to the fact that it takes a longer time to find intersections on surfaces by step tracking. However, this approach has many beneficial properties, such as robustness and determinism. This has been demonstrated in various data experiments and is even faster in some cases.

For completeness, we also recorded the time required for feature extraction, with DoG + HardNet taking 0.21 ms per feature and DoG + HardNet + AffNet taking 0.64 ms. The results are shown in Table 8.

Table 8. Average time of GC-RANSAC using DG + AN + HN on Cambridge Landmarks and Aachen Day-Night datasets.

Scene			Time (s) ↓		
	P3P	**P1AC**	**UP2P**	**IPPE**	**TLS**
Great Court	**0.14**	0.26	0.17	0.21	0.28
King's Col.	**0.19**	0.48	0.18	0.37	0.47
Old Hospital	**0.12**	0.41	0.41	0.38	0.45
Shop Façade	**0.13**	0.38	0.21	0.36	0.40
St Mary's Ch.	**0.12**	0.38	0.72	0.29	0.37
Street	**0.11**	0.29	0.22	0.27	0.31
Day	0.23	**0.11**	0.14	0.18	0.13
Night	0.13	0.10	0.11	**0.06**	0.11

4.3. Navigation Application Effect

Using the motion tracking technology that fuses the camera and IMU data as the basis, the A* algorithm does the navigation path planning to realize the AR indoor navigation system application. The application is deployed to Redmi K50 cell phones to test the application effect in a design exhibition hall.

Figure 7 shows the process of feature matching to obtain four PCs. First, we mark and measure the key points that need to be navigated in the scene, consider the posters in the scene as markers, make the four corners of the poster as reference points, record their positions, and store this information in the database.

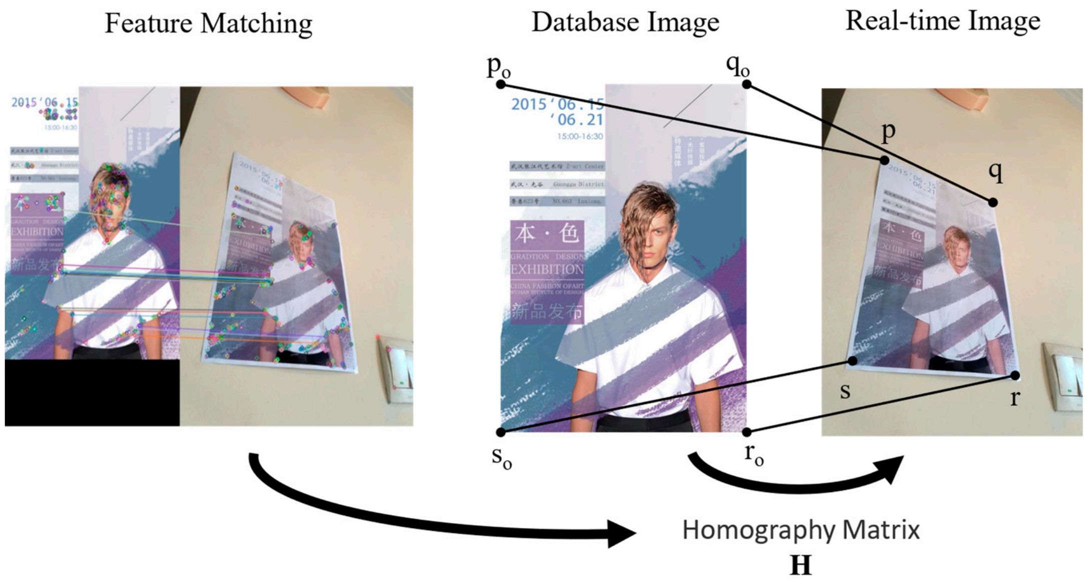


Figure 7. The process of feature matching to obtain four PCs.

By using the feature point pairs obtained from Harris corner detection and ORB operator matching, we were able to obtain the homograph ***H*** between the poster in the database and in the application scene. Through ***H***, we were able to determine four PCs of the poster corners. Combined with the TLS visual localization, we corrected the camera's position and thus adjusted the pointing of the navigation route. The correction effect is shown in Figure 8.

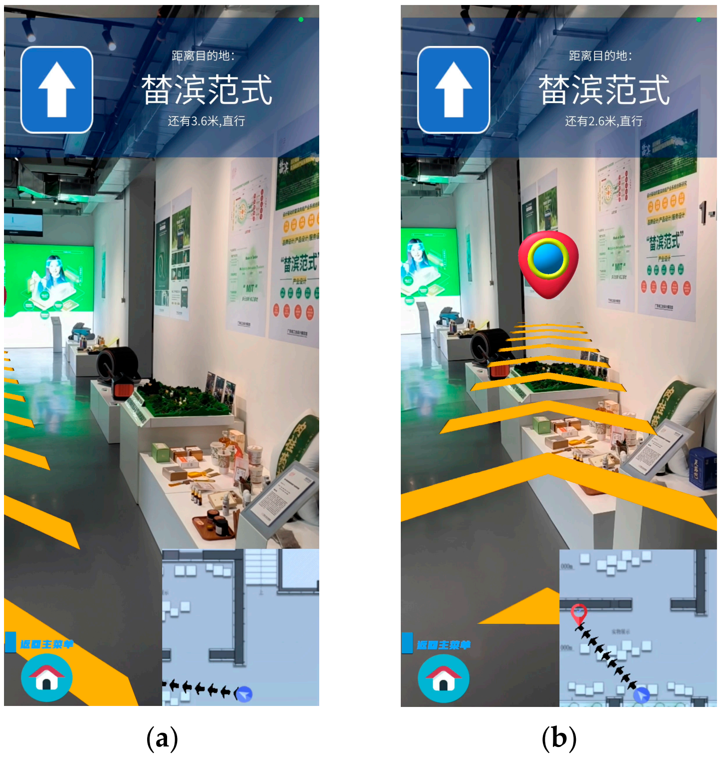


Figure 8. The path navigation performance of the AR indoor navigation system. (**a**) Without TLS visual localization; (**b**) with TLS visual localization.

Figure 8 shows the path navigation effect of the AR indoor navigation system. Figure 8a represents the navigation effect obtained only by relying on the motion tracking technology, which does not allow the arrow path to accurately guide to the "菩滨范式" destination due to the cumulative error when moving; Figure 8b represents the effect of real-time correction of the device's position by combining with the TLS visual localization in the system, which enables the device to accurately perceive its own positioning in the scene, thus forming a more accurate navigation route to the destination, reflecting the accuracy of TLS.

We also record ten sets of key localization points and connect them into paths to compare the accuracy of TLS. As shown in Figure 9, the background image is an aerial view cross-section of the pavilion, the paths consisting of diamond-shaped points are ground truth (GT) measured in the scene, and the paths consisting of triangular and square points are visual localization with/without TLS, respectively.

As can be seen from the aerial view, relying only on the motion tracking of the mobile device will make the localization have a great deal of randomness, thus seriously affecting the quality of the navigation. After applying the visual localization algorithm in this paper, the obtained navigation paths can better fit the pre-set navigation routes, reflecting the practicality of TLS.

The superposition of AR navigation paths and video images is another segmentation theme of the navigation guide project, which uses the A* algorithm to plan a feasible path based on the camera position and the target position and real-time visualization of the path and the target point in Unity, which is discussed in a separate contribution by the author due to the long length of the algorithm.

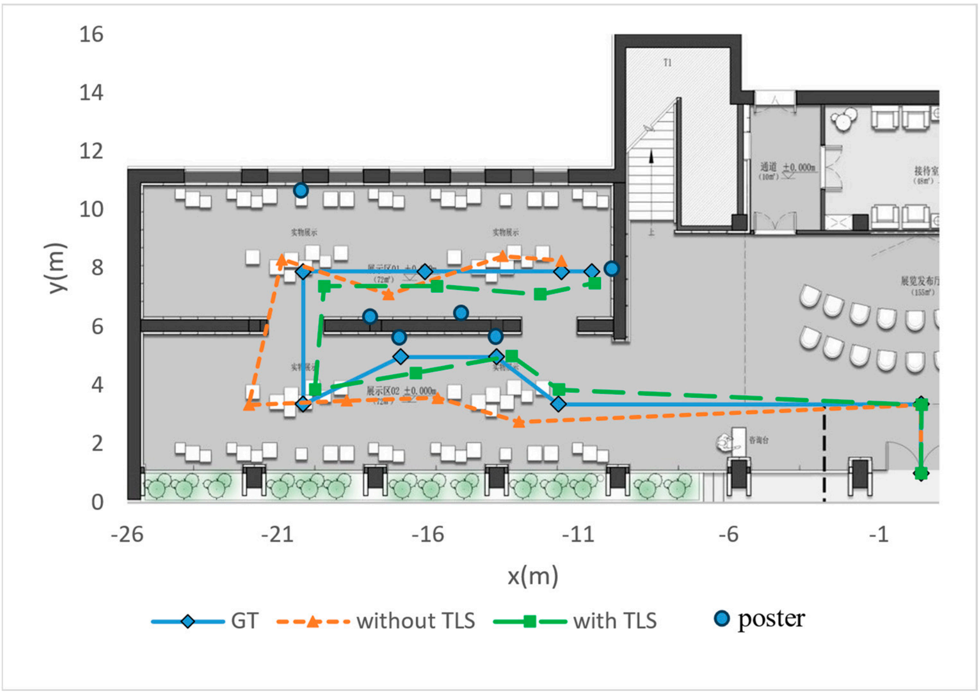


Figure 9. Aerial view of the navigation path.

5. Conclusions

In this paper, the camera's absolute position is used as the entry point. Three torus-like surface geometric constraints are constructed by combining the four PCs with the camera intrinsics and focal length. The problem of solving the position of four PCs is transformed into the surface intersection. We utilize the step-size adaptive tracking algorithm to quickly find the available intersections and then solve the rotation matrix. Experiments show that our method has excellent performance in terms of stability, time, and accuracy. Its accuracy and usefulness in navigation application testing are demonstrated. Future work will be centered around constructing three torus-like surfaces with fewer PCs. We will also explore what geometric constraints to apply in this situation to obtain the optimal positional solution.

Author Contributions: Conceptualization, L.L.; methodology, L.L. and X.L.; software, X.L.; validation, L.L., X.L., and B.S.; formal analysis, L.L. and X.L.; investigation, L.L., X.L. and B.S.; resources, L.L. and X.L.; data curation, B.S.; writing—original draft preparation, X.L.; writing—review and editing, L.L., X.L. and B.S.; visualization, X.L. and B.S.; supervision, L.L.; project administration, L.L.; funding acquisition, L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Guangdong Province, China (No. 2023A1515011706), and the Humanities and Social Sciences Project of the Ministry of Education of China (No. 20YJAZH073).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Si, S.B.; Zhao, D.W.; Xu, W.Y. Review on visual-inertial navigation and positioning technology. *J. Image Graph.* **2021**, *26*, 13. [CrossRef]
2. Zhang, H.; Guo, M.; Zhao, W.; Huang, J.; Meng, Z.; Lu, P.; Sen, L.; Sheng, B. Visual indoor navigation using mobile augmented reality. In Proceedings of the Computer Graphics International Conference, Virtual Event, 12–16 September 2022; pp. 145–156.
3. Huang, G.; Cai, H.; Deng, C.; He, Z.; Xu, N. A Visual Localization Method Based on Indoor Signs. *J. Transp. Inf. Saf.* **2021**, *39*, 172–179.

4. Zhang, T.; Xu, Y.; Feng, L.; Xue, Z. Research on Indoor Visual Positioning System Based on QR Code. *Acta Opt. Sin.* **2024**, *44*, 0915001.
5. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
6. Barath, D.; Matas, J. Graph-cut RANSAC: Local optimization on spatially coherent structures. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 4961–4974. [CrossRef]
7. Haralick, B.M.; Lee, C.-N.; Ottenberg, K.; Nölle, M. Review and analysis of solutions of the three point perspective pose estimation problem. *Int. J. Comput. Vis.* **1994**, *13*, 331–356. [CrossRef]
8. Collins, T.; Bartoli, A. Infinitesimal plane-based pose estimation. *Int. J. Comput. Vis.* **2014**, *109*, 252–286. [CrossRef]
9. Thompson, E.H. Space resection: Failure cases. *Photogramm. Rec.* **1966**, *5*, 201–207. [CrossRef]
10. Köser, K.; Koch, R. Differential spatial resection-pose estimation using a single local image feature. In Proceedings of the European Conference on Computer Vision, Marseille, France, 12–18 October 2008; pp. 312–325.
11. Sweeney, C.; Flynn, J.; Nuernberger, B.; Turk, M.; Höllerer, T. Efficient computation of absolute pose for gravity-aware augmented reality. In Proceedings of the 2015 IEEE International Symposium on Mixed and Augmented Reality, Fukuoka, Japan, 29 September–3 October 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 19–24.
12. Ventura, J.; Kukelova, Z.; Sattler, T.; Baráth, D. P1ac: Revisiting absolute pose from a single affine correspondence. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–6 October 2023; pp. 19751–19761.
13. Li, R.; Lu, L.; Jin, G. Research Overview of Location Method for Monocular Vision. *Mod. Comput.* **2011**, *11*, 9–12.
14. Gui, Y.; Lao, S.; Kang, L.; Liang, B. Accuracy Assessment Method of PnP Algorithm in Visual Geo-localization. *Comput. Sci.* **2018**, *45*, 13–16+27.
15. Ke, T.; Roumeliotis, S.I. An efficient algebraic solution to the perspective-three-point problem. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7225–7233.
16. Persson, M.; Nordberg, K. Lambda Twist: An accurate fast robust perspective three-point (P3P) solver. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 318–332.
17. Ding, Y.; Yang, J.; Larsson, V.; Olsson, C.; Åström, K. Revisiting the P3P problem. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 4872–4880.
18. Haring, M.; Grøtli, E.I.; Riemer-Sørensen, S.; Seel, K.; Hanssen, K.G. A Levenberg-Marquardt algorithm for sparse identification of dynamical systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 9323–9336. [CrossRef] [PubMed]
19. Mannel, F.; Aggrawal, H.O.; Modersitzki, J. A structured L-BFGS method and its application to inverse problems. *Inverse Probl.* **2024**, *40*, 045022. [CrossRef]
20. Zhuang, S.; Zhao, Z.; Cao, L.; Wang, D.; Fu, C.; Du, K. A robust and fast method to the perspective-n-point problem for camera pose estimation. *IEEE Sens. J.* **2023**, *23*, 11892–11906. [CrossRef]
21. Jiang, C.; Hu, Q. Iterative pose estimation for a planar object using virtual sphere. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 3650–3657. [CrossRef]
22. Mills, S. Four-and seven-point relative camera pose from oriented features. In Proceedings of the 2018 International Conference on 3D Vision (3DV), Verona, Italy, 5–8 September 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 218–227.
23. Barath, D. Five-point fundamental matrix estimation for uncalibrated cameras. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 235–243.
24. Kukelova, Z.; Bujnak, M.; Pajdla, T. Closed-form solutions to minimal absolute pose problems with known vertical direction. In Proceedings of the Asian Conference on Computer Vision, Queenstown, New Zealand, 8–12 November 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 216–229.
25. Chen, L.; Gong, Y.; Li, B.; Ren, X.; Yang, C.; Wang, X. Algorithm to Quickly Calculate the Intersecting Lines Between Free-Form Surfaces in Cad/Cam. *J. Mech. Eng.* **2000**, *36*, 102–105. [CrossRef]
26. Li, J. On the definition of equivalence of close planes. *J. Sci. Teachers' Coll. Univ.* **1995**, *15*, 7–9.
27. Liu, P.; He, X. Research on Surface Intersection Technology Based on Step-Size Adaptive Tracking Method. *Comput. Eng. Appl.* **2020**, *56*, 253–259.
28. Lin, H. Survey on Geometric Iterative Methods with Applications. *J. Comput.-Aided Des. Comput. Graph.* **2015**, *27*, 582–589. [CrossRef]
29. Ventura, J.; Kukelova, Z.; Sattler, T.; Baráth, D. Absolute Pose from One or Two Scaled and Oriented Features. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–22 June 2024; pp. 20870–20880.
30. Eichhardt, I.; Barath, D. Relative pose from deep learned depth and a single affine correspondence. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part XII 16. Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 627–644.
31. Barath, D.; Hajder, L. Novel ways to estimate homography from local affine transformations. In Proceedings of the Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), Rome, Italy, 27–29 February 2016; pp. 434–445.
32. Lebeda, K.; Matas, J.; Chum, O. Fixing the Locally Optimized RANSAC. In Proceedings of the British Machine Vision Conference, Surrey, UK, 3–7 September 2012.

33. Kendall, A.; Grimes, M.; Cipolla, R. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2938–2946.
34. Zhang, Z.; Sattler, T.; Scaramuzza, D. Reference pose generation for long-term visual localization via learned features and view synthesis. *Int. J. Comput. Vis.* **2021**, *129*, 821–844. [CrossRef]
35. Wu, C. Towards linear-time incremental structure from motion. In Proceedings of the IEEE International Conference on 3D Vision, Seattle, WA, USA; 2013; pp. 127–134.
36. Schonberger, J.L.; Frahm, J.M. Structure-from-motion revisited. In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4104–4113.
37. Lowe, D.G. Distinctive image features from scaleinvariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
38. Mishkin, D.; Radenovic, F.; Matas, J. Repeatability is not enough: Learning affine regions via discriminability. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 284–300.
39. Mishchuk, A.; Mishkin, D.; Radenovic, F.; Matas, J. Working hard to know your neighbor's margins: Local descriptor learning loss. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4826–4837.
40. DeTone, D.; Malisiewicz, T.; Rabinovich, A. SuperPoint: Self-supervised interest point detection and description. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–23 June 2018; pp. 224–236.
41. Lindenberger, P.; Sarlin, P.E.; Pollefeys, M. Lightglue: Local feature matching at light speed. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 17627–17638.
42. Lee, J.; Kim, B.; Cho, M. Self-supervised equivariant learning for oriented keypoint detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 4847–4857.