Fast Globally Optimal Surface Normal from an Affine Correspondence

Levente Hajder, Lajos Lóczi
Geometric Computer Vision Group
Eötvös Loránd University, Budapest, Hungary
{hajder,lloczi}@inf.elte.hu

Daniel Barath
Computer Vision and Geometry Group
ETH Zurich, Switzerland
dbarath@inf.ethz.ch

Abstract

We present a new solver for estimating a surface normal from a single affine correspondence in two calibrated views. The proposed approach provides a new globally optimal solution for this over-determined problem and proves that it reduces to a linear system that can be solved extremely efficiently. This allows for performing significantly faster than other recent methods, solving the same problem and obtaining the same globally optimal solution. We demonstrate on 15k image pairs from standard benchmarks that the proposed approach leads to the same results as other optimal algorithms while being, on average, five times faster than the fastest alternative. Besides its theoretical value, we demonstrate that such an approach has clear benefits, e.g., in image-based visual localization, due to not requiring a dense point cloud to recover the surface normal. We show on the Cambridge Landmarks dataset that leveraging the proposed surface normal estimation further improves localization accuracy. Matlab and C++ implementations are also published in the supplementary material.

1. Introduction

Acquiring oriented point clouds is a fundamental problem both in computer vision and in robotics with a number of important real-world applications benefiting from the extra information it provides. Such applications include 3D reconstruction [1, 8, 31, 62, 72, 65, 56] and simultaneous localization and mapping [19, 23, 50], where the 3D point orientations provide useful information about the underlying structure. Image-based visual localization approaches [43, 53, 60, 61] can also benefit from known surface normals [68]. The widely used Poisson reconstruction [37, 38] is based on both the point coordinates and their orientations. The visual odometry [51, 52] of an autonomous vehicle can also benefit from understanding surface orientation to, *e.g.*, segment ground or facades. Also, having an oriented point cloud greatly simplifies geometric model and multi-model estimation [32, 4, 5] due to allowing to design solvers that require fewer data points than their point-based counterparts [64]. Moreover, normals can also be leveraged in the numerical refinement of the reconstructed oriented 3D point cloud [27]. In this paper, we focus on a practical problem, *i.e.*, estimating the normal at corresponding features in two images in a solely photometric manner. This can provide a *lightweight* approach of obtaining surface normals without requiring a dense point cloud to be known. Next, we will discuss ways of obtaining oriented points.

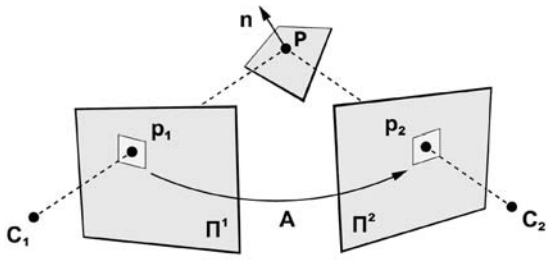

Figure 1. **Geometric configuration:** oriented 3D point $\mathbf{P} \in \mathbb{E}^3$ with normal $\mathbf{n} \in \mathbb{E}^3$ projected to image Π^1 as $\mathbf{p}_1 \in \mathbb{E}^2$ and to Π^2 as $\mathbf{p}_2 \in \mathbb{E}^2$. The local patch centered at points $\mathbf{p}_1$ and $\mathbf{p}_2$ are related by the local affine transformation $\mathbf{A} \in \mathbb{R}^{2\times2}$.

Normals from Point Clouds. After applying state-of-the-art sparse (*e.g.*, COLMAP [63]) or dense [36, 16] reconstruction pipelines, the resulting point cloud can be straightforwardly equipped with point orientations. In this case, typically, the normals are fitted per-point, applying the widely used Principal Component Analysis (PCA) [35] to estimate tangent planes or higher-order surfaces [14, 33] from the local neighborhood centered at the current point. This procedure, while being frequently used, has two major drawbacks. First, the setting of the neighborhood size greatly affects the accuracy and is often unclear without having a metric reconstruction. Instead, using the k-nearest-neighbors [17] algorithm to find local neighborhoods is a viable solution, but the estimation becomes largely affected by the sparsity of the point cloud. Second, fitting local surfaces, independently, to the neighborhood of each 3D point

in the reconstruction is an extremely expensive operation.

Normals from Homographies. Another method for estimating surface normals from images, without requiring the underlying 3D structure to be known, is via homographies. Homographies are projective transformations between images of co-planar 3D points that depends on the intrinsic and extrinsic camera parameters and the parameters of the underlying 3D plane [24]. A homography can be estimated from a minimum of four point correspondences in two images [30]. While this method allows the estimation of dominant planes supported by a number of correspondences in the images, it does not provide means for normal estimation for each local feature correspondence independently.

Normals from Local Mappings. In recent years, affine correspondences (AC) have emerged as a popular topic in computer vision, offering a valuable source of information about the underlying 3D geometry of the surrounding environment. An AC consists of a corresponding pair of points in two images and a linear transformation that maps the infinitesimal region around the point in the first image to the point in the second one. They have been used to solve various geometric problems, including relative pose estimation in the general case [15, 55, 3, 22, 10], for known vertical direction [25], planar motion [25, 26], or known feature depths [21] and homography estimation [2].

Traditionally, finding affine features involve four main steps: (scale-covariant) keypoint detection, orientation and affine shape estimation, and descriptor extraction. Under such paradigm, keypoint detection is typically performed on the scale pyramid with the help of a handcrafted response function, such as Hessian [13, 45], Harris [28, 45], Difference of Gaussians (DoG [42]), or learned ones like FAST [57] or Key.Net [11]. Keypoint detection obtains a triplet (x, y, scale) which defines a square or circular image patch. Then the patch orientation is estimated with the help of handcrafted – dominant gradient orientation [42], center of the mass [58] – or learned methods [71, 48, 41]. Next, the affine-covariant shape [12, 48] is estimated. Finally, the patch is geometrically rectified and fed into a local patch descriptor, such as SIFT, HardNet [47], SOSNet [66], or others. In summary, all feature detectors, even recent learning-based ones like SuperPoint [20], can be made affine covariant by applying, *e.g.*, AffNet [48] as a post-processing step on the found features.

The problem of surface normal estimation from ACs has been addressed in several works. The first algorithm for normal estimation from ACs was proposed by Kevin Koeser [40] assuming that the camera location is known with respect to the observed 3D point. Megyesi et al. [44] proposed a special surface normal estimator for standard (rectified) stereo, where the degree of freedom of the affine transformations decreases to two. In 2014, Molnár and Chetverikov [49] derived a formula connecting the surface normals, camera parameters, affine transformations, and patch locations into a single expression for arbitrary projective functions, like the catadioptric camera model. Barath et al. [9] proposed a globally optimal estimator in the least squares sense as the solution of a quartic polynomial. Recently, Le Minh and Hajder [46] improved on this by reducing the degree and showed that the optimal solution can be found as a real root of a cubic polynomial.

The main goal of this paper is to propose a rapid normal estimator based on affine correspondences. Our algorithm can be inserted into a 3D reconstruction system, such as Structure-from-Motion [27] (SfM) or Simultaneous Localization and Mapping (SLAM), where the normal estimation is performed many times, therefore its running time influences the time demand of the whole 3D pipeline.

Contribution. In this paper, we propose a new *linear* and globally *optimal* solution for surface normal estimation from a single affine correspondence. The proposed solver requires only the four basic arithmetic operations and contains 19 parameters, consisting of the coordinates of the five input vectors and the four affine parameters. This approach completely avoids root extractions, solving high-degree polynomial equations, or employing numerical or iterative methods, resulting in a particularly fast algorithm that runs for only 0.6μs in our experiments. The solver is *five* times faster than the cubic solver from [46]. Additionally, we propose a way of exploiting AC-wise normals in image-based visual localization pipelines. The algorithms are tested on 15k image pairs from standard benchmarks, and our results show superior accuracy and faster wall-clock times compared to the baselines.

2. Proposed Method

The setting is visualized in Fig 1. We are given two images, denoted by Π^1 and Π^2. The 2D and 3D point locations are visualized as black dots. For standard PC-based stereo vision, the 3D location can be obtained by the so-called triangulation method. The local affine transformation, denoted by $\mathbf{A} \in \mathbb{R}^{2\times 2}$, is defined as the first-order Taylor-approximation of the imaging function at points $\mathbf{p}_1$ and $\mathbf{p}_2$. It maps the infinitesimal patch around the observed points from the first to the second image. If $\mathbf{A}$ is known, the surface normal $\mathbf{n}$ can be directly estimated without knowing the actual 3D structure of the underlying scene [49].

Parameterization. A surface normal $\mathbf{n} \in \mathbb{R}^3$ is a 3D direction, therefore it can be represented by two parameters. Usually, it is parameterized by three coordinates $\mathbf{n} = [n_x, n_y, n_z]^\top$ and its length being set to 1. However, as it turns out below, some other scaling will be more convenient in the present situation.

Problem Statement. Let us assume that we are given a general camera that projects the spatial location $\mathbf{P} = [X, Y, Z]^\top$ to image (pixel) coordinates $[u\ v]^\top$, and let us

denote the projective coordinate functions by $\Pi_u : \mathbb{E}^3 \to \mathbb{R}$ and $\Pi_v : \mathbb{E}^3 \to \mathbb{R}$. Function Π_u calculates coordinate u of the projected point, and Π_v calculates v. As Molnár and Chetverikov showed [49], the affine transformation can be defined via the spatial gradients of the projective coordinate functions Π_u and Π_v.

An affine correspondence is represented by 2D image coordinates $[u_1 \quad v_1]^\top$ and $[u_2 \quad v_2]^\top$ in the stereo image pair, in conjunction with the affine transformation $\mathbf{A}$ that maps the surrounding area of the locations from first image to the second one. The spatial location of the corresponding spatial point is also required. It can be estimated from the image coordinates and camera parameters by standard triangulation techniques [29].

The normal estimation problem is written [9] as an argument minimization to determine the optimal $\mathbf{n}^*$ such that

$$\mathbf{n}^* = \arg\min_{\mathbf{n}} f(\mathbf{n}),$$

with

$$f(\mathbf{n}) := \sum_{i=1}^{2}\sum_{j=1}^{2}\left(\frac{\mathbf{n}^\top \mathbf{w}_{ij}}{\mathbf{n}^\top \mathbf{w}_c} - a_{ij}\right)^2. \tag{1}$$

Here, the affine transformation yields the four affine parameters a_{ij} ($i, j \in [1, 2]$) according to

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

and the vectors $\mathbf{w}_{ij}$ and $\mathbf{w}_c$ are obtained as cross products of the spatial gradient vectors as follows:

$$\begin{array}{ll} \mathbf{w}_{11} = \nabla\Pi_v^1 \times \nabla\Pi_u^2, & \mathbf{w}_{12} = \nabla\Pi_u^2 \times \nabla\Pi_u^1, \\ \mathbf{w}_{21} = \nabla\Pi_v^1 \times \nabla\Pi_v^2, & \mathbf{w}_{22} = \nabla\Pi_v^2 \times \nabla\Pi_u^1, \\ \mathbf{w}_c = \nabla\Pi_u^1 \times \nabla\Pi_v^1, & \end{array}$$

where the superscripts refer to the corresponding image index. The coordinates of these vectors will be denoted by $\mathbf{w}_{ij,x}, \mathbf{w}_{ij,y}, \mathbf{w}_{ij,z}$ ($i, j \in \{1, 2\}$) and $\mathbf{w}_{c,x}, \mathbf{w}_{c,y}, \mathbf{w}_{c,z}$.

Pin-hole (perspective) projection model. For a pin-hole camera, the projection model is as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

where $\sim$ denotes the equality up to scale operation. Therefore, image coordinates u and v are obtained as

$$\Pi_u = \frac{P_{11}X + P_{12}Y + P_{13}Z + P_{14}}{P_{31}X + P_{32}Y + P_{33}Z + P_{34}},$$
$$\Pi_v = \frac{P_{21}X + P_{22}Y + P_{23}Z + P_{24}}{P_{31}X + P_{32}Y + P_{33}Z + P_{34}}.$$

Spatial gradients, that are $\nabla\Pi = \left[\begin{array}{ccc} \frac{\partial\Pi}{\partial X} & \frac{\partial\Pi}{\partial Y} & \frac{\partial\Pi}{\partial Z} \end{array}\right]^\top$, can be written for the pin-hole model in compact forms as

$$\nabla\Pi_u = \frac{1}{s}\begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \end{bmatrix}^\top - \frac{u}{s}\begin{bmatrix} P_{31} \\ P_{32} \\ P_{33} \end{bmatrix}^\top,$$
$$\nabla\Pi_v = \frac{1}{s}\begin{bmatrix} P_{21} \\ P_{22} \\ P_{23} \end{bmatrix}^\top - \frac{v}{s}\begin{bmatrix} P_{31} \\ P_{32} \\ P_{33} \end{bmatrix}^\top,$$

where $s = P_{31}X + P_{32}Y + P_{33}Z + P_{34}$ is the projective depth.

Solution. For the problem to be meaningful, we can assume that $\mathbf{w}_c \neq \mathbf{0}$. Let us set

$$I := \inf_{\|\mathbf{n}\|=1,\ \mathbf{n}^\top\mathbf{w}_c \neq 0} f(\mathbf{n}),$$

where $\|\mathbf{n}\| := \sqrt{\mathbf{n}^\top\mathbf{n}}$ denotes the standard Euclidean norm. Clearly, $f \geq 0$ is undefined along the "equator" of the unit sphere $\|\mathbf{n}\| = 1$ with $\mathbf{n}^\top\mathbf{w}_c = 0$. Let us consider two cases.

If all $\mathbf{w}_{ij}$ and $\mathbf{w}_c$ are equal, then trivially $f(\mathbf{n}) = \sum_{i,j=1}^{2}(a_{ij} - 1)^2$ is a constant function.

Otherwise, at least one of the expressions $\frac{\mathbf{n}^\top\mathbf{w}_{ij}}{\mathbf{n}^\top\mathbf{w}_c}$ becomes arbitrarily large in modulus when $\mathbf{n}$ is chosen such that $\mathbf{n}^\top\mathbf{w}_c$ gets sufficiently close to 0; that is, $|f(\mathbf{n})| \to +\infty$ as $\mathbf{n}^\top\mathbf{w}_c \to 0$ with $\|\mathbf{n}\| = 1$. Therefore, by taking into account the continuity of f as well, the infimum I is in fact a minimum, and an optimal unit vector $\mathbf{n}^*$ with $f(\mathbf{n}^*) = I$ exists. In what follows, we will explicitly determine such an $\mathbf{n}^*$ as a function of the 19 parameters [1] by using only the four basic arithmetic operations.

A key observation is that $f(\lambda\mathbf{n}) = f(\mathbf{n})$ for any $\lambda \neq 0$, so (after a possible rescaling) we can assume that the last component of $\mathbf{n}$ is either 1 or 0. These cases, Cases A and B, respectively, will be discussed separately below. To simplify the notation, let us define two auxiliary functions

$$g(n_x, n_y) := f(n_x, n_y, 1) \tag{2}$$

and

$$h(n_x, n_y) := f(n_x, n_y, 0). \tag{3}$$

For the objective function g, we will obtain a unique candidate extremum $\mathbf{n}_g^* \in \mathbb{R}^2$; whereas for the function h, a one-parameter family of candidate extrema $\mathbf{n}_h^* \in \mathbb{R}^2$ will be found but with the same function value $h(\mathbf{n}_h^*)$. The optimal $\mathbf{n}^*$ is then

$$\mathbf{n}^* = \begin{cases} [\mathbf{n}_g^*, 1]^\top, & \text{if } g(\mathbf{n}_g^*) \leq h(\mathbf{n}_h^*), \\ [\mathbf{n}_h^*, 0]^\top, & \text{if } g(\mathbf{n}_g^*) > h(\mathbf{n}_h^*). \end{cases} \tag{4}$$

[1] They are the four affine parameters a_{ij}, and the 15 components of the vectors $\mathbf{w}_{ij}$ and $\mathbf{w}_c$.

Remark 1 *In other words, we will be utilizing the simple constraints $n_z = 1$ or $n_z = 0$. We have found that other linear or non-linear constraints, such as $\|\mathbf{n}\| = 1$, would render the symbolic calculations to minimize the objective function f infeasible.*

Case A: $n_z = 1$

We use the standard technique and solve the system

$$\begin{cases} \partial_1 g(n_x, n_y) = 0, \\ \partial_2 g(n_x, n_y) = 0 \end{cases} \tag{5}$$

to find the optimal vector $\mathbf{n}_g^* = [n_x, n_y]^\top$ as follows.

By computing and putting the partial derivative

$$\partial_1 g(n_x, n_y)$$

over a common denominator, we see that its numerator depends on n_x linearly; and it depends on n_y quadratically; so one can easily express n_x from the equation $\partial_1 g(n_x, n_y) = 0$ as follows:

$$n_x = \frac{\text{numer}_1}{\text{denom}_1}, \tag{6}$$

where we present the expressions for numer_1 and denom_1 in the supplementary material. Then (6) is substituted into the other partial derivative $\partial_2 g(n_x, n_y)$. The resulting rational function of n_y, also containing the other 19 parameters, can be simplified and factorized by Wolfram *Mathematica*'s `Factor` command [2]. In this way, we obtain a fraction whose structure is the following:

$$\frac{\text{linear}_1 \cdot \text{linear}_2}{(\text{quadratic})^2}, \tag{7}$$

where the expressions "linear" and "quadratic" denote polynomials in n_y having the corresponding degrees.

The optimum of g can occur when n_y is chosen such that the fraction (7), that is, its numerator is equal to 0. It can be shown that the assumption $\text{linear}_1 = 0$ leads to a contradiction – indeed, by solving $\text{linear}_1 = 0$ for n_y, the resulting formula would make denom_1 in (6) identically equal to 0. Hence, we are only left with the equation $\text{linear}_2 = 0$, whose solution is

$$n_y = \frac{\text{numer}_2}{\text{denom}_2}. \tag{8}$$

The actual formulas for numer_2 and denom_2 are found again in the supplementary material.

Thus, the second component of $\mathbf{n}_g^*$ is given by (8) (expressed in terms of the 19 parameters). Then, by using this together with (6), we obtain n_x, the first component of $\mathbf{n}_g^*$.

[2] https://www.wolfram.com/mathematica/

Remark 2 *The essential feature of these computations is that the solution of system* (5) *is given in a fully explicit symbolic form by using only elementary operations. Without the clever intermediate symbolic simplifications and factorizations (most notably* (7)*), one would need to solve a higher-degree polynomial equation to obtain n_y, requiring either taking roots or using an iterative scheme. For example, if one starts solving* (5) *with n_x and n_y as symbolic parameters but all the coefficients of g as floating-point numbers, then, in general, no intermediate factorizations or simplifications are possible, so in order to determine n_y, one would need to solve a quintic equation.*

Case B: $n_z = 0$

We proceed similarly and solve the system

$$\begin{cases} \partial_1 h(n_x, n_y) = 0, \\ \partial_2 h(n_x, n_y) = 0. \end{cases} \tag{9}$$

By expressing n_x from the first equation of (9) and substituting the resulting expression into the second equation, we find that this time the second equation is automatically satisfied. Therefore, (9) possesses a one-parameter family of solutions given by

$$n_x = \frac{\text{numer}_3}{\text{denom}_3}, \tag{10}$$

see the supplementary material. By substituting this into h, the function value $h(\mathbf{n}_h^*)$ appearing in (4) is obtained as

$$h(\mathbf{n}_h^*) = \frac{\text{numer}_4}{\text{denom}_4}. \tag{11}$$

See the supplementary material for details.

Remark 3 *One directly checks that the value $h(\mathbf{n}_h^*) = h\left(\frac{\text{numer}_3}{\text{denom}_3}, n_y\right)$ is indeed independent of n_y, as claimed.*

2.1. Summary of the Algorithm

Given the elements $a_{ij} \in \mathbb{R}$ in the affine transformation and 3D vectors $\mathbf{w}_{ij}$ and $\mathbf{w}_c \neq \mathbf{0}$, let us define the function f by (1) for $\mathbf{n}$ with $\mathbf{n}^\top \mathbf{w}_c \neq 0$, and also define the functions g and h by (2)–(3). If $\mathbf{w}_{11} = \mathbf{w}_{12} = \mathbf{w}_{21} = \mathbf{w}_{22} = \mathbf{w}_c$, then any unit vector $\mathbf{n}^*$ can be chosen as optimal normal vector.

Otherwise, we compute $\mathbf{n}_g^* := [n_x, n_y]^\top$ from (8) and (6). Then we compute $h(\mathbf{n}_h^*)$ from (11). Finally, the optimal $\mathbf{n}^*$ is determined by using (4).

3. Experiments

We focus on the accuracy and time demand of the algorithms when testing. Both synthetic testing data, including configurations for a number of important camera motions, and standard real-world benchmarks are considered here.

Figure 2. Images and the reference reconstruction used from the Cambridge Landmarks dataset [39] in localization experiments.

Compared Algorithms. In this section, we discuss the algorithms that can be used to estimate the surface normal from a single affine correspondence.

1. The provably most accurate algorithms are the ones that are optimal in the least squares sense. Method $\text{OPT}_{\text{QUARTIC}}$ refers to [9] which solves the problem as a real root of a quartic polynomial equation. The cubic solution $\text{OPT}_{\text{CUBIC}}$ is from [46]. The proposed optimal solver, as it is solved via linear equations, is called $\text{OPT}_{\text{LINEAR}}$ in the rest of the paper.
2. The simplest and, thus, the fastest method is the Fast Normal Estimation (FNE) algorithm of [9]. A more complex variant, when all possible linear equations are considered, is also constructed here, it is denoted as $\text{FNE}_{\text{SUPER}}$. The modified equations can be seen in the source codes in the supplementary material.
3. Raposo and Barreto [56] proposed a solution that estimates the normal via homographies. We have not found an implementation or an exact definition of the algorithm, so we implemented it based on the description. The solution is computed via an over-determined inhomogeneous linear system of equations. The formulas are generated by the MATLAB Symbolic Toolbox. Details are given in the supplementary material. The method is called RAPOSO in the charts.

The synthetic tests are performed in MATLAB. In the real-world experiments, optimized C++ codes run on an Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz.

3.1. Synthetic Experiments

For the synthetic tests, a spherical object is generated, thus straightforwardly allowing to sample oriented points from its surface. Two cameras are randomly generated with focal lengths from the range $[500, 1500]$ and with image size 2000×2000. Their intrinsic matrices are $\mathbf{K}_1$ and $\mathbf{K}_2$. The 3D point was projected into both cameras. The ground truth (GT) affine parameters were calculated directly from the camera parameters and the surface normals as shown in [9]. Several camera configurations are considered to simulate various real-world scenarios. They are as follows:

- For **general stereo pose**, the locations of the cameras are randomly generated inside a unit-radius sphere. The camera projection matrices are constructed as $\mathbf{P}_1 = \mathbf{K}_1[\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = \mathbf{K}_2[\mathbf{R} \mid \mathbf{t}]$.
- For **rectified stereo**, the cameras are located next to each other with identical orientation. In this case, the projection matrices are $\mathbf{P}_1 = \mathbf{K}_1[\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = \mathbf{K}_2[\mathbf{I} \mid \mathbf{t}]$, where $\mathbf{t} = [\Delta X\ 0\ 0]^\top$.
- **Planar motion** simulates the case when a camera is mounted on a vehicle that moves on a planar road with the image plane orthogonal to the ground. Such a motion can be represented by two angles around the vertical axis, one for the rotation and one for the translation. In this case, the projection matrices can be written as $\mathbf{P}_1 = \mathbf{K}_1[\mathbf{I} \mid 0]$ and $\mathbf{P}_2 = \mathbf{K}_2[\mathbf{R}_\text{Y} \mid \mathbf{t}]$, where $\mathbf{t} = \rho[\cos\beta, 0, \sin\beta]^\top$ and

$$\mathbf{R}_\text{Y} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}.$$

- **Moto-motion.** This test case simulates a camera that is mounted on a motorcycle. If the bike leans, another rotation, the roll is involved in the problem. In this case, the camera parameters are $\mathbf{P}_1 = \mathbf{K}_1[\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = \mathbf{K}_2[\mathbf{R}_\text{YZ} \mid \mathbf{t}]$, where $\mathbf{t} = \rho[\cos\beta\ 0\ \sin\beta]^\top$ and $\mathbf{R}_{YZ} = \mathbf{R}_Y\mathbf{R}_Z$.
- **Drone-motion** is another generalization of the planar motion. A camera mounted on a flying drone. Its location is arbitrary, but the orientation is limited to a vertical rotation. Therefore, $\mathbf{P}_1 = \mathbf{K}_1[\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = \mathbf{K}_2[\mathbf{R}_\text{Y} \mid \mathbf{t}]$, where $\mathbf{t} = [X\ Y\ Z]^\top$.

Noise. We have added zero-mean Gaussian noise to the 2D coordinates, therefore, the point noise is given in pixels. The deviation of the noise is the parameter in the charts.

The 2×2 affine transformations are decomposed by SVD into two 2D rotations and another two scaling parameters. Then zero-mean Gaussian noise is added to the angle of rotations, in radians, and the scaling parameters. The standard deviation of the Gaussian noise distribution is equally set for all the four parameters.

3D Reconstruction Pipeline. In order to simulate real scenarios, first, we triangulate each generated noisy point correspondence by the optimal triangulation method proposed by Hartley and Sturm [29]. The noisy affine parameters are corrected by the method proposed in [7] to be consistent with the camera parameters.

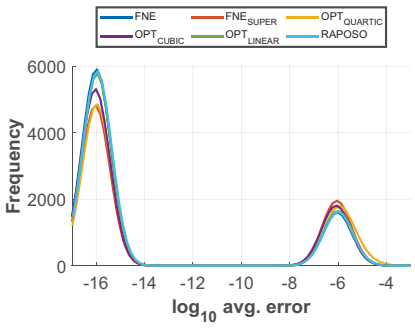

Figure 3. The frequencies (in 10000 runs) of the $\log_{10}$ errors in the $\log_{10}$ angular errors in degrees in the noise-free case.

Stability test. We generated 10000 random problem instances considering general camera motion and ran the normal estimators on noiseless affine correspondences. Fig. 3 shows histograms of $\log_{10}$ angular errors in degrees. Note that for better visualization, we replace the exact zero values by 10^{-16} as it is the precision of the floating point numbers in MATLAB. The plots show that all our solvers are very stable – there is no peak close to 10^0. For all methods, the majority of the errors are distributed around 10^{-16} which actually is the precision of MATLAB. This means that in most of the cases, all methods are particularly stable returning almost exactly 0 error. A small peak is over 10^{-6} for all methods similarly. Having 10^{-6} degree error is acceptable in most of the scenarios considered in computer vision. There is no peak close to 10^0 implying that all solvers are stable. Note that these stability tests usually are for highlighting potential degeneracies where the error exceeds 10^0.

We also conducted tests using single-point precision in MATLAB. The $\text{OPT}_{\text{QUARTIC}}$ approach failed entirely under these conditions. For all other methods, we noticed an increase in error, which suggests that double precision is necessary for reliable normal estimation. Interestingly, our proposed $\text{OPT}_{\text{LINEAR}}$ method appears to be slightly more sensitive to precision than $\text{OPT}_{\text{CUBIC}}$ is. This is likely due to the fact that $\text{OPT}_{\text{LINEAR}}$ employs more complex formulas and, consequently, performs more stably with double precision.

Accuracy test. The accuracy of the normals is examined for all the considered camera poses independently. The results are shown in Fig 9. The effect of noise in point coordinates and affine parameters are separately evaluated. Both the mean and the median of the errors are plotted.

The observations we can make from the plots are as follows. First, the three optimal methods $\text{OPT}_{\text{QUARTIC}}$, $\text{OPT}_{\text{CUBIC}}$ and $\text{OPT}_{\text{LINEAR}}$ always lead exactly to the same solution, thus their curves coincide in all plots. Second, the optimal methods always lead to the most accurate results if the affine parameters are contaminated by noise. This is not surprising as these methods optimize the normal parameters such that the implied affine transformation is the closest to the measured one. In the case of point-noise only, the accuracy of the optimal algorithm is slightly below that of the competitors. However, it is an unlikely situation in real-world images to have noisy points but perfect local affine frames. It is usually the other way around. Third, as the FNE method uses less information, it is clearly the worst one in terms of accuracy. $\text{FNE}_{\text{SUPER}}$ performs significantly better, however, it requires more time as the right singular vector of a 6×3 matrix, corresponding to the smallest singular value, should be computed. Another benefit of $\text{FNE}_{\text{SUPER}}$ is, contrary to FNE and Raposo [56], that it works in all test cases. It can cope with both planar motion and rectified stereo, similarly to the optimal algorithms.

Problematic methods. Methods FNE and Raposo [56] do not work if the cameras undergo special translations. FNE is a special method, several variants can be formed based on the selection of the affine transformations. For rectified stereo, the third and fourth elements are fixed [44], they cannot be used for the estimation.

The Raposo [56] method is the most problematic one as the homography can be decomposed into the camera rotation, translation and surface normal. If the translation has zero elements, the retrieved equations are linearly dependent, and the estimation does not yield a correct solution. Due to this reason, Raposo [56] does not work for rectified stereo and planar motion. We removed these methods from the plots of the respective camera configurations.

Combined noise. In the previous experiments, noise is added either to the point coordinates or affine parameters. Here, we contaminate both types of input data. The mean and median errors of the proposed linear method are visualized in Fig. 10. As it is expected, the curves show similar trends, but the case when both the points and affines are noisy leads to higher errors.

Interpretable affine noise. In the previous experiments, the noise is added directly to the elements of the affine matrices. In Fig. 11, we consider a different way of contaminating the input to show that the proposed method acts reasonably with different noise models as well. The results of curves *Aff* and *Comb* are obtained by using the noise model from the previous sections. *Comb* refers to the case when both the points and the affine matrices are noisy. For *Aff-Elem* and *Comb-Elem*, we decompose $\mathbf{A}$ by SVD into two orthogonal and a diagonal matrix. Fig. 11 demonstrates that the noise generation does not strongly influence the results.

4. Real-world Experiments

To test normal estimators, we use the Cambridge Landmarks [39] and ScanNet datasets [18]. We chose Cambridge Landmarks, see Fig. 2 for example images, since it provides a large number of images for testing directly the

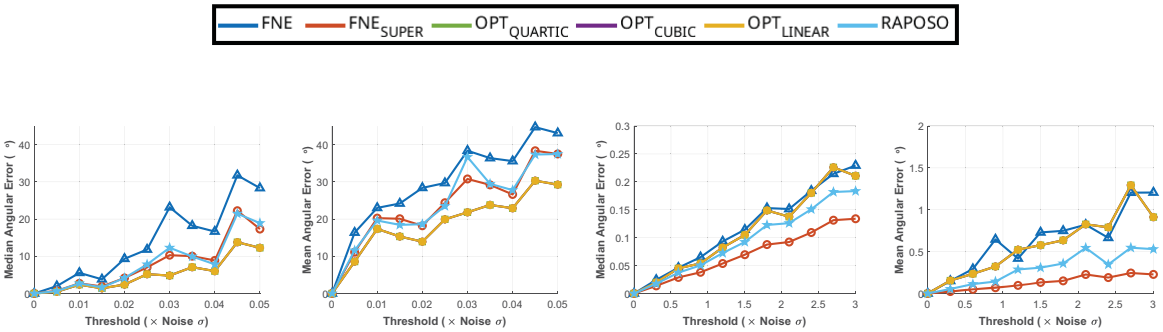

Figure 4. General Motion.

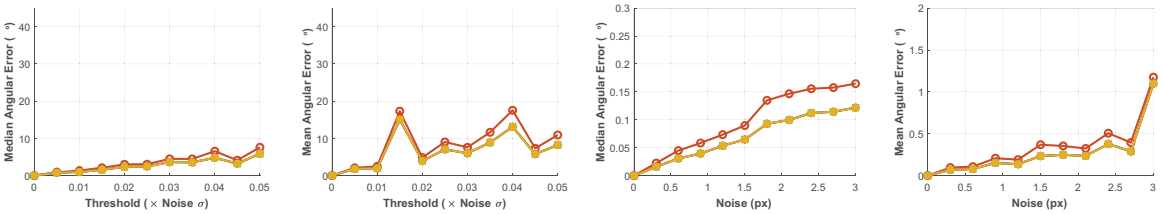

Figure 5. Rectified Stereo.

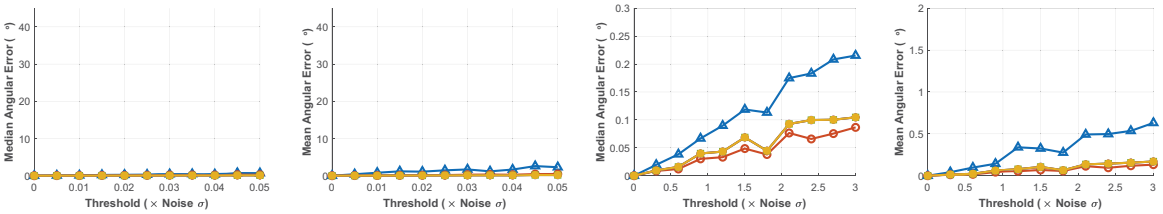

Figure 6. Planar Motion.

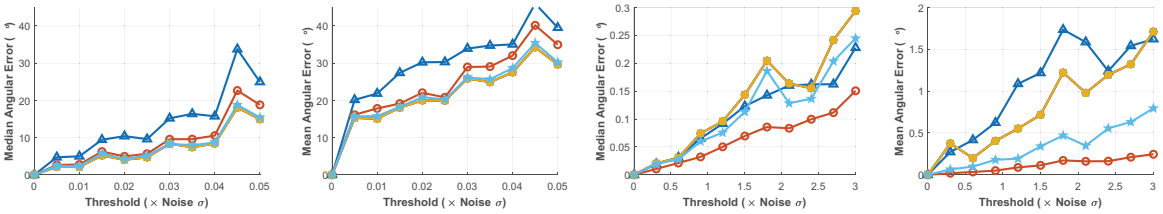

Figure 7. Moto Motion.

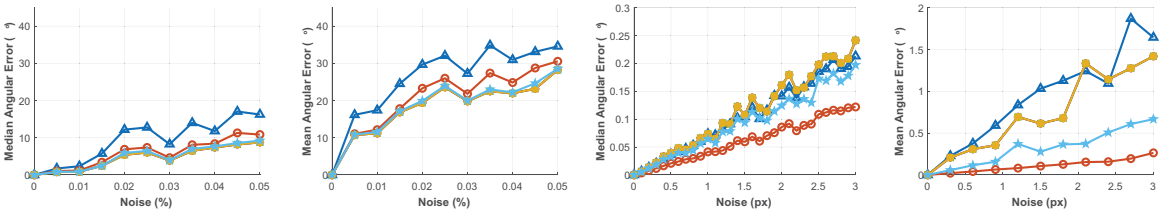

Figure 8. Drone Motion.

Figure 9. Average and median angular errors (in degrees) of surface normal estimators on synthetic data. Five camera configurations are considered. From left to right: (1) Median error; noise added to affine parameters. (2) Average error; noise added to affine parameters. (3) Median error; noise added to point coordinates. (4) Average error; noise added to point coordinates.

normal accuracy and, also, we can evaluate localization as an application of the proposed procedure. Also, we chose ScanNet since the reference normals can be calculated directly from the dense depth maps which provides a complementary comparison to the tests on Cambridge Landmarks, where the normals come from a 3D reconstruction.

Cambridge Landmarks consists of five scenes with varying number of images, ranging from 231 to 3015. The reference pose of each image was reconstructed using the VisualSFM structure-from-motion software [70, 69] and the

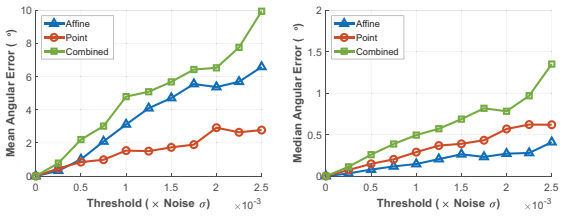

Figure 10. Mean and median errors of the proposed $\text{OPT}_{\text{LINEAR}}$ normal estimator with noise added to point coordinates (*Point*), affine parameters (*Affine*) or both (*Combined*).

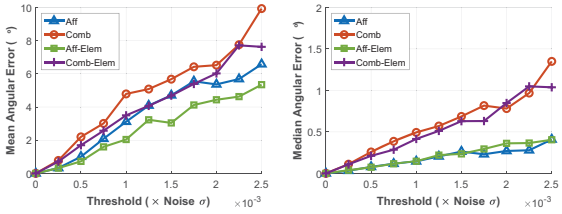

Figure 11. Mean and median errors of the proposed $\text{OPT}_{\text{LINEAR}}$ normal estimator with noise added to only the affine (*Aff*) elements and, also, to the point coordinates (*Comb*). For *Aff* and *Comb*, zero-mean Gaussian noise is added directly to the affine elements. For *Aff-Elem* and *Comb-Elem*, the affine matrix is decomposed by SVD to two rotations and a diagonal matrix. Zero-mean Gaussian noise is added to the rotation angles and diagonal elements.

images are divided into training and testing sets. To have more accurate reference data, we ran COLMAP [63] obtaining a dense point cloud. Finally, we applied the normal estimator implemented in MeshLab[3] to obtain an oriented point cloud to be considered reference. The normals are fitted to the 200 nearest neighbors. For each query image, we ran DenseVLAD [67] to get 20 potential database images. In our experiments, the algorithms are tested on each obtained image pair, *i.e.*, 20 pairs for each query image. For ScanNet, we use the 1500 image pairs used in [59].

On Cambridge Landmarks, we detect affine correspondences by detecting DoG features [42], finding the affine shape by AffNet [48], and extracting HardNet [47] descriptors. This approach is among the leaders in the IMC 2020 benchmark [34]. On ScanNet, we detect SuperPoint [20] features, match them by SuperGlue [59], and find the affine shape by, first, estimating scale and orientation by SelfScaleOri [41] and, then, finding the affine shapes by AffNet.

Normal Estimation. To test normal estimators, we ran all of them on the data described in the previous section. For each image pair, we iterate through all detected correspondences that are consistent with the camera pose – they have less than 5 pixels of Sampson distance. For each of them, we ran multiple normal estimators, measure their angular error and run-time. To understand how the estimators proceed with respect to the error in the detected affine transformations $\mathbf{A}$, we calculate the reference transformation $\mathbf{A}^*$ from the camera parameters and the surface normal as shown in [9]. We define the affine error as the Frobenius norm of the difference matrix $||\mathbf{A} - \mathbf{A}^*||_{\text{F}}$.

[3]Meshlab is an open source tool for 3D point cloud processing and visualization.

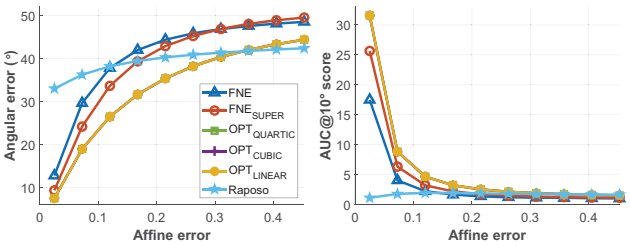

Figure 12. The median angular error in degrees (left) and the AUC@10° score (right) of the estimated normals plotted as a function of the error in the detected affine correspondences on the Cambridge Landmarks dataset.

The Area Under the recall Curve (AUC) at 5°, 10° and 20°, the avg. and med. angular errors with the standard deviation in degrees and the run-time in nanoseconds on the Cambridge Landmarks dataset are reported in Table 1. As expected, the best results are achieved by the optimal algorithms, all of them leading to exactly the same results. Our proposed linear solution of the optimal problem, $\text{OPT}_{\text{LINEAR}}$, is *five* times faster than the cubic solution $\text{OPT}_{\text{CUBIC}}$. While $\text{OPT}_{\text{LINEAR}}$ is slightly slower than the fastest FNE method, it is significantly more accurate. Interestingly, while the method of Raposo et al. [56] worked well in the synthetic experiments, we did not manage to achieve good accuracy with it in the real-world experiments. Figure 12 plots the AUC@10° score and the median angular error as a function of the noise in the detected affine parameters. As expected, normal estimation is highly sensitive to the quality of the affine parameters. The optimal algorithms are the most stable ones.

The results on the ScanNet dataset are reported in Table 3. Similarly as on Cambridge Landmarks, all optimal solvers lead to the same accuracy while the proposed one being the fastest by a large margin.

Visual Localization. To test the impact of surface normal estimation on localization, on the Cambridge Landmarks dataset, we applied the following procedure. We iterate through all query images, each having 10 database images retrieved by DenseVLAD. First, we estimate the relative pose between the query and a random database image by GC-RANSAC [6] with P3P [54]. Using the obtained pose, we estimate the surface normal of each inlier affine correspondence in the coordinate system of the query image. Next, we iterate through all unused database images and use

Estimator	AUC@5° ↑	@10° ↑	@20° ↑	Avg. err. (°) ↓	Med. err. (°) ↓	Std. dev. (°) ↓	Time (ns) ↓
FNE [49]	6.0	16.0	33.7	21.0	14.3	21.3	**172**
$\text{FNE}_{\text{SUPER}}$	8.9	22.1	42.0	10.9	16.8	15.6	1721
Raposo [56]	0.4	1.5	5.7	38.5	36.5	18.2	883
$\text{OPT}_{\text{QUARTIC}}$ [9]	**11.7**	**27.3**	**48.7**	**8.8**	**13.6**	**12.9**	2528
$\text{OPT}_{\text{CUBIC}}$ [46]	**11.7**	**27.3**	**48.7**	**8.8**	**13.6**	**12.9**	2484
Proposed $\text{OPT}_{\text{LINEAR}}$	**11.7**	**27.3**	**48.7**	**8.8**	**13.6**	**12.9**	596

Table 1. The AUC score at 5°, 10°, and 20° of the estimated surface normals, their average and median errors in degrees, the standard deviation, and the average run-time, in nanoseconds, of a single normal estimation on the Cambridge Landmarks dataset.

Solver	Normal estimator	Pose accuracy ↑ AUC@1°, 10cm	@2.5°, 25cm	@5°, 50cm	Med. rot. err. (°) ↓	Med. pos. err. (cm) ↓
P3P	–	44.3	59.1	67.5	10.1	13.5
P1AC	$\text{FNE}_{\text{SUPER}}$	47.8	62.0	69.7	6.2	12.8
P1AC	Proposed $\text{OPT}_{\text{LINEAR}}$	**48.0**	**62.3**	**69.9**	**5.2**	**1.8**
P1AC	GT	48.4	62.9	70.4	4.6	2.1

Table 2. Localization AUC scores at (1°, 10cm), (2.5°, 25cm), and (5°, 50cm), the median rotation (in degrees) and translation errors (in centimeters) on the Cambridge Landmarks dataset [39] when using either the P3P solver or the normal-based P1AC solver with $\text{FNE}_{\text{SUPER}}$ or the proposed surface normal estimator $\text{OPT}_{\text{LINEAR}}$. We also show the results with the reference normals. The minimal solvers are used within Graph-Cut RANSAC [6] that takes care of the robust estimation.

	AUC@5 ↑	AUC@10 ↑	AUC@20 ↑	Time (ns) ↓
FNE	14.8	15.4	17.0	207
$\text{FNE}_{\text{SUPER}}$	34.9	35.2	36.6	2142
$\text{OPT}_{\text{QUARTIC}}$	**35.0**	**35.5**	**37.3**	18745
$\text{OPT}_{\text{CUBIC}}$	**35.0**	**35.5**	**37.3**	3146
$\text{OPT}_{\text{LINEAR}}$	**35.0**	**35.5**	**37.3**	**747**

Table 3. The AUC scores at 5°, 10° and 20°, and the avg. time (in nanoseconds) of normal estimators on the ScanNet datasets [18].

	P3P	P1AC GT	P1AC $\text{FNE}_{\text{SUPER}}$	P1AC Prop. $\text{OPT}_{\text{LINEAR}}$
St. Mary's Church	44.2	47.7	46.1	**46.5**
Old Hospital	44.1	57.4	56.8	**57.3**
King's College	52.1	54.2	**54.1**	54.0
Shop Facade	65.5	65.0	63.8	**64.1**
Great Court	62.7	63.2	63.0	**63.5**
Street	17.4	22.8	**23.1**	23.0

Table 4. The area under the recall curve (AUC) at 1° rotation and 10cm position errors on the Cambridge Landmarks dataset. We combine the P1AC solver with $\text{FNE}_{\text{SUPER}}$ and the proposed normal estimator. Also, we test it with the reference normals.

the P1AC [68] solver inside GC-RANSAC to estimate the pose. P1AC requires a single AC and the surface normal. We only test the proposed solver and $\text{FNE}_{\text{SUPER}}$. All other optimal solvers lead exactly to the same accuracy as ours. $\text{FNE}_{\text{SUPER}}$ is the most accurate sub-optimal algorithm.

The accuracy of the estimated pose for the database images with respect to their GT pose is reported in Table 2. All solvers, benefiting from normals and affine correspondences, are more accurate than using P3P. We also report the accuracy when using the reference normals to see the upper bound achievable using normals. The proposed optimal solver $\text{OPT}_{\text{LINEAR}}$ is only slightly (*i.e.* $<$ 1 AUC point) behind the accuracy of the GT normal, while being more accurate than the P3P-based solution. The AUC@10° scores on each scene are reported in Table 4. The proposed $\text{OPT}_{\text{LINEAR}}$ leads to the best accuracy on all but two scenes.

5. Conclusions

In this work, we have presented a novel solver for estimating surface normals from affine correspondences when having calibrated cameras. The proposed method yields the optimal solution, in the least squares sense, with respect to the affine error, and is particularly fast as the optimal normal is obtained via linear equations. We compared our method to all existing alternatives that estimate normals from a single correspondence on both simulated and real-world data. Additionally, we provide tests that highlight the benefits of using surface normals in image-based localization. The proposed estimator has demonstrated superior accuracy compared to the fastest existing method while being the second fastest overall. Our results suggest that the proposed algorithm is a promising alternative for surface normal estimation in computer vision applications.

Acknowledgement. The authors warmly thank the reviewers for their very valuable comments and suggestions. The work was supported by the ETH Postdoc fellowship.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Commun. ACM*, 54(10), 2011.

[2] Daniel Barath and Levente Hajder. A theory of point-wise homography estimation. *Pattern Recognit. Lett.*, 94:7–14, 2017.

[3] Daniel Barath and Levente Hajder. Efficient recovery of essential matrix from two affine correspondences. *IEEE Transactions on Image Processing*, 27(11):5328–5337, 2018.

[4] Daniel Barath and Jiri Matas. Multi-class model fitting by energy minimization and mode-seeking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 221–236, 2018.

[5] Daniel Barath and Jiri Matas. Progressive-x: Efficient, anytime, multi-model fitting algorithm. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3780–3788, 2019.

[6] Daniel Barath and Jiri Matas. Graph-cut RANSAC: local optimization on spatially coherent structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4961–4974, 2021.

[7] Daniel Barath, Jiri Matas, and Levente Hajder. Accurate closed-form estimation of local affine transformations consistent with the epipolar geometry. In *BMVC*, 2016.

[8] Daniel Barath, Dmytro Mishkin, Ivan Eichhardt, Ilia Shipachev, and Jiri Matas. Efficient initial pose-graph generation for global sfm. In *CVPR*, 2021.

[9] D. Barath, J. Molnar, and L. Hajder. Novel methods for estimating surface normals from affine transformations. In *Computer Vision, Imaging and Computer Graphics Theory and Applications, Selected and Revised Papers*, pages 316–337. Springer International Publishing, 2015.

[10] Daniel Barath, Michal Polic, Wolfgang Förstner, Torsten Sattler, Tomas Pajdla, and Zuzana Kukelova. Making affine correspondences work in camera geometry computation. In *European Conference on Computer Vision*, pages 723–740. Springer, 2020.

[11] Axel Barroso-Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters. In *ICCV*, 2019.

[12] Adam Baumberg. Reliable feature matching across widely separated views. In *CVPR*, pages 1774–1781. IEEE Computer Society, 2000.

[13] P. R. Beaudet. Rotationally invariant image operators. In *Proceedings of the 4th International Joint Conference on Pattern Recognition*, 1978.

[14] Pál Benkö, Géza Kós, Tamás Várady, László Andor, and Ralph R. Martin. Constrained fitting in reverse engineering. *Comput. Aided Geom. Des.*, 19(3):173–205, 2002.

[15] Jacob Bentolila and Joseph M Francos. Conic epipolar constraints from affine correspondences. *Computer Vision and Image Understanding*, 122:105–114, 2014.

[16] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1538–1547, 2019.

[17] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13:21– 27, 1967.

[18] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[19] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Toward geometric deep SLAM. *CoRR*, 2017.

[20] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[21] Ivan Eichhardt and Daniel Barath. Relative pose from deep learned depth and a single affine correspondence. In *European Conference on Computer Vision*, pages 627–644. Springer, 2020.

[22] Iván Eichhardt and Dmitry Chetverikov. Affine correspondences between central cameras for rapid relative pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 482–497, 2018.

[23] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: large-scale direct monocular SLAM. In *ECCV*, 2014.

[24] Olivier D. Faugeras and Francis Lustman. Motion and structure from motion in a piecewise planar environment. *Int. J. Pattern Recognit. Artif. Intell.*, 2(3):485–508, 1988.

[25] Banglei Guan, Ji Zhao, Zhang Li, Fang Sun, and Friedrich Fraundorfer. Minimal solutions for relative pose with a single affine correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1929–1938, 2020.

[26] Levente Hajder and Daniel Barath. Relative planar motion for vehicle-mounted cameras from a single affine correspondence. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8651–8657. IEEE, 2020.

[27] Levente Hajder and Ivan Eichhardt. Computer vision meets geometric modeling: Multi-view reconstruction of surface points and normals using affine correspondences. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2427–2435, 2017.

[28] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[29] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding: CVIU*, 68(2):146–157, 1997.

[30] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[31] Jared Heinly, Johannes L. Schönberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the world* in six days. In *CVPR*, 2015.

[32] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International journal of computer vision*, 97(2):123–147, 2012.

[33] Xiangmin Jiao and Duo Wang. Reconstructing high-order surfaces for meshing. *Eng. Comput.*, 28(4):361–373, 2012.

[34] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision*, 2020.

[35] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374, 2016.

[36] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NeurIPS*, 2017.

[37] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics*, pages 61–70, 2006.

[38] Michael Kazhdan and Hugues Hoppe. Screened Poisson surface reconstruction. *ACM TOG*, page 29, 2013.

[39] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[40] Kevin Köser and Reinhard Koch. Differential Spatial Resection - Pose Estimation Using a Single Local Image Feature. In *ECCV*, pages 312–325, 2008.

[41] Jongmin Lee, Yoonwoo Jeong, and Minsu Cho. Self-supervised learning of image scale and orientation. In *31st British Machine Vision Conference 2021, BMVC 2021, Virtual Event, UK*. BMVA Press, 2021.

[42] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[43] Simon Lynen, Bernhard Zeisl, Dror Aiger, Michael Bosse, Joel A. Hesch, Marc Pollefeys, Roland Siegwart, and Torsten Sattler. Large-scale, real-time visual-inertial localization revisited. *IJRR*, 39(9), 2020.

[44] Z. Megyesi, G. Kos, and D. Chetverikov. Dense 3D Reconstruction from Images by Normal Aided Matching. *Machine Graphics & Vision*, 15:3–28, 2006.

[45] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision (IJCV)*, 60(1):63–86, 2004.

[46] Nghia Le Minh and Levente Hajder. Affine transformation from fundamental matrix and two directions. In *VISAPP*, pages 819–826, 2020.

[47] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working Hard to Know Your Neighbor's Margins: Local Descriptor Learning Loss. In *NeurIPS*, 2017.

[48] D. Mishkin, F. Radenovic, and J. Matas. Repeatability is Not Enough: Learning Affine Regions via Discriminability. In *ECCV*, 2018.

[49] József Molnár and Dmitry Chetverikov. Quadratic transformation for planar mapping of implicit surfaces. *J. Math. Imaging Vis.*, 48(1):176–184, 2014.

[50] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics*, 31(5), 2015.

[51] David Nistér, Oleg Naroditsky, and James R. Bergen. Visual odometry. In *CVPR*, 2004.

[52] David Nistér, Oleg Naroditsky, and James R. Bergen. Visual odometry for ground vehicle applications. *J. Field Robotics*, 23(1), 2006.

[53] Vojtech Panek, Zuzana Kukelova, and Torsten Sattler. Meshloc: Mesh-based visual localization. In *ECCV*, 2022.

[54] Mikael Persson and Klas Nordberg. Lambda Twist: An accurate fast robust perspective three point (p3p) solver. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 318–332, 2018.

[55] Carolina Raposo and Joao P Barreto. Theory and practice of structure-from-motion using affine correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5470–5478, 2016.

[56] Carolina Raposo and João Pedro Barreto. Accurate reconstruction of oriented 3d points using affine correspondences. In *ECCV*, volume 12373, pages 545–560. Springer, 2020.

[57] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag.

[58] E. Rublee, V. Rabaud, K. Konolidge, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *ICCV*, 2011.

[59] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[60] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012.

[61] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomás Pajdla. Benchmarking 6dof outdoor visual localization in changing conditions. In *CVPR*, 2018.

[62] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[63] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[64] Agnes Seiler, David Großmann, and Bert Jüttler. Spline surface fitting using normal data and norm-like functions. *Comput. Aided Geom. Des.*, 64:37–49, 2018.

[65] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *IJCV*, 80(2), 2008.

[66] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. Sosnet: Second order similarity regularization for local descriptor learning. In *CVPR*, 2019.

[67] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1808–1817, 2015.

[68] Jonathan Ventura. P1ac: Revisiting absolute pose from a single affine correspondence. *arXiv preprint arXiv:2011.08790*, 2020.

[69] Changchang Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 127–134. IEEE, 2013.

[70] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *CVPR 2011*, pages 3057–3064. IEEE, 2011.

[71] K. M. Yi, Y. Verdie, P. Fua, and V. Lepetit. Learning to Assign Orientations to Feature Points. In *CVPR*, 2016.

[72] Siyu Zhu, Runze Zhang, Lei Zhou, Tianwei Shen, Tian Fang, Ping Tan, and Long Quan. Very large-scale global sfm by distributed motion averaging. In *CVPR*, 2018.