Efficient Large Scale Inlier Voting for Geometric Vision Problems

Dror Aiger aigerd@google.com
Simon Lynen slynen@google.com
Jan Hosang hosang@google.com
Bernhard Zeisl bzeisl@google.com
Google Research

Abstract

Outlier rejection and, equivalently, inlier set optimization is a key ingredient in numerous applications in computer vision such as filtering point-matches in camera pose estimation or plane and normal estimation in point clouds. Several approaches exist, yet at large scale we face a combinatorial explosion of possible solutions and state-of-the-art methods like RANSAC, Hough transform, or Branch&Bound require a minimum inlier ratio or prior knowledge to remain practical. In fact, for problems such as camera posing in very large scenes these approaches become useless as they have exponential runtime growth.

To approach the problem, we present an efficient and general algorithm for outlier rejection based on "intersecting" k-dimensional surfaces in R^d. We provide a recipe for formulating a variety of geometric problems as finding a point in R^d which maximizes the number of nearby surfaces (and thus inliers). The resulting algorithm has linear worst-case complexity with a better runtime dependency on the requested proximity of a query to its result than competing algorithms, while not requiring domain specific bounds. This is achieved by introducing a space decomposition scheme that bounds the number of computations by successively rounding and grouping surfaces. Our recipe and open-source code[1] enables anybody to derive such fast approaches to new problems across a wide range of domains. We demonstrate the approach on several camera posing problems with a large number of matches and low inlier ratio, achieving state-of-the-art results at significantly lower processing times.

1. Introduction

Whether geometric verification of point matches for absolute pose, homography estimation or normal estimation in point clouds, outlier rejection is a key ingredient in numerous applications in computer vision. RANSAC [8] is a common choice for the task as it has good quality/runtime tradeoffs for many practical problems. RANSAC however does not provide an optimality guarantee, requires a lower bound on the inlier ratio and its runtime increases exponentially with the outlier ratio, making it unusable for problems with few inliers. In contrast Branch&Bound methods [3, 6] are guaranteed to find the optimal solution, yet their worst-case runtime equals exhaustive search in parameter space. Their practical use depends on the quality of the bounds which are problem specific and notoriously hard to find.

We propose an alternative approach that exhibits a linear runtime independent of the inlier ratio and that does not require a lower bound on the fraction of inliers. We base our work on the efficient and general algorithm for solving geometric incidence problems proposed by a subset of the authors in [1]. In a nutshell the authors show that voting [23] is equivalent to finding a point in $\mathbb{R}^d$ which is close to as many surfaces (and thus inliers) as possible among a given set of k-surfaces in $\mathbb{R}^d$. While in [1] we proves theoretical bounds for approximate incidences, we show a general scheme to formulate approximate incidences of surfaces in a unified algorithm and apply it to several inlier maximization problems. We obtain a *globally optimal solution* (up to a prescribed resolution ε) considerably faster. Our contributions:

- We introduce the concept of *general voting* for outlier removal and its relation to our previous work on *approximate incidences* in [1] to a wider computer vision audience and demonstrate its use for camera posing, ray intersection, and geometric model fitting.
- All approaches have *linear* complexity and are therefore applicable to very large problems that are infeasible with RANSAC [8]. The worst case complexity (and performance in practice) is always better than other methods (voting [11] and B&B [14]) for sufficiently large inputs.
- We demonstrate the scalability and versatility using a generalization of [23, 21] where we remove the requirement of calibrated cameras and known gravity direction, but instead solve for these unknowns.
- We compare to the state-of-the-art [6] for 6 degree-of-

[1] https://github.com/google-research/google-research/tree/master/large_scale_voting

freedom (6DoF) camera pose estimation and show that our approach is optimal and faster in practice. In comparison to [6] our surface intersection algorithm provides a tight upper bound on the score without problem specific knowledge.

– We provide example-derivations and open-source code which serve as a tutorial for applying [1] to a range of outlier removal problems in computer vision.

2. The Family of General Alignment Problems

A large number of geometric vision problems can be viewed as general alignment problems in which we aim to bring items in set a A "close" to items in another set B by applying a transformation from a group of allowed transformations. Closeness is defined by some parameter $\varepsilon > 0$ and a distance metric, commonly the Hausdorff [12] distance between items in A and B. *Note that ε defines which items we consider close enough, which is part of the problem definition — ε is* **not** *an approximation factor.* [2].

Many outlier rejection problems can be framed as such an alignment problem. For example in structure-based localization [17, 19] A is a set of rays in 3-space in the camera frame that we want to align with a set of 3d world points B. In relative camera posing [9, 10] we align two sets of 3-rays such that the maximum number of pairs are ε-close. In all of these problems, one defines the items in A and B and the allowed transformation group and then solves an approximate geometric incidence problem.

Any hypothetical match — a correspondence between an object in A and an object in B — defines a *general surface* that is embedded in the ambient d-space of transformations. Therefore, we can solve the outlier rejection in linear time in the number of hypothetical matches by a voting scheme: surfaces are "ε-intersected" in the d-space to locate the point close to most surfaces. This point represents the solution to the alignment problem.

3. Proximity and Incidences using Surfaces

Our work is based on the findings of [1] and we introduce their approach and notation using 2D line fitting as a toy example. The authors formulate the maximum incidence problem as one of reporting points in $\mathbb{R}^d$ that are close to the maximal number of k-dimensional surfaces. Each surface $\sigma \in S$ is given in parametric form where the first k coordinates $\mathbf{x} = (x_1, \ldots, x_k)$ are the surface parameters. Specifically, each σ is defined in terms of ℓ *essential parameters* $\mathbf{t} = (t_1, \ldots, t_\ell)$, and $d-k$ additional *free additive parameters* $\mathbf{f} = (f_{k+1}, \ldots, f_d)$, one free parameter for each dependent coordinate.[3] The surface σ is parameterized by $\mathbf{t}$ and $\mathbf{f}$ (we then denote σ as $\sigma_{\mathbf{t},\mathbf{f}}$) and defined by

$$\begin{aligned} &F^{(\sigma)} : \mathbb{R}^k \times \mathbb{R}^\ell \to \mathbb{R}^{(d-k)} \\ &x_j = F_j^{(\sigma)}(\mathbf{x}; \mathbf{t}) + f_j, \qquad \text{for } j = k+1, \ldots, d. \end{aligned} \tag{1}$$

Without loss of generality the voting space is scaled to be the unit cube $[0,1]^d$ to simplify runtime complexity expressions. Unbounded parameters are either still bounded in practice or can be transformed in the problem parameterization as is common in B&B (e.g. $\cos(\theta)$ for line fitting).

For the toy example of 2D line fitting, we are given a set of points in 2D to which we want to fit a 2D line: We want to find the line that is ε-close to the maximum number of points. In order to formulate this problem within our surface proximity scheme we use standard duality [2] which maps points to lines and lines to points. This allows us to find a point in 2D which is ε-close to the maximum number of points instead. Each input point is mapped into a $k = 1$ dimensional surface (a line) embedded in the $\mathbb{R}^{d=2}$ line parameter space. The surface equation in this case is

$$x_2 = a \cdot x_1 + b = F_2^{(\sigma)}(x_1; a) + b, \tag{2}$$

where the only essential parameter is the slope $\mathbf{t} = (a)$ and the only free parameter is the offset $\mathbf{f} = (b)$. Consequently, the point in dual space $\mathbb{R}^{d=2}$ with the most close surfaces corresponds to the line parameters (in the primal space) that fit the input points best.[4]

3.1. A Naive Voting Solution

Once we formulate a problem as a general surface consensus problem, we obtain a naive Algorithm 1:

Iterate the first k dimensions of the voting space on an ε-grid and compute the dependent variables $(x_{k+1}, ..., x_d)$ for each surface in an 3ε neighborhood box around each grid vertex. That is, for each vertex we collect all nearby surfaces at most ε apart from each other. Then use the dependent variable range to cast votes in the voting space. For higher dimensional spaces this enumeration becomes costly, since the algorithm is independent of the actual structure of the surfaces and always takes $O(n/\varepsilon^k)$ time for n k-dimensional surfaces.

3.2. Efficient Canonized Generalized Voting

Algorithm 2 is our generalized voting procedure based on the work in [1]. We propose a coarse-to-fine scheme

[2] Our algorithm is globally optimal (up to a prescribed resolution if we want linear time) and for simplicity we set the resolution to be also ε throughout the paper

[3] We separate essential from free parameters in notation, since they will be handled differently in the algorithm that follows.

[4] Technically, we introduce some error, since duality preserves only vertical distances between lines and points, not Euclidean distances. We keep the error bounded by separating lines into two groups: lines with slope $a' \in [-1, 1]$ and lines with the inverse slope $a' = 1/a = [-1, 1]$, resulting in the two voting spaces (x_1, x_2) and (x_2, x_1) that have a 1-to-1 correspondence for each point in the space. When searching a maximum, we have to consider the sum of both spaces.

Algorithm 1: Naive Voting

Data: S: surfaces, B: box in $[0,1]^d$, ε: distance
Result: Point in B (among all ε-grid points) with the maximum of ε-close surfaces.
for *all* $s \in S$ **do**
  for *all* k*-dimensional* ε*-cells in* B **do**
    Compute the $d-k$ dependent variables using the surface equations.
    Tally a vote for the d-tuple and add s into the set of inliers corresponding to it.
    /* The d-tuple is the concatenation of the k-tuple and the dependent $d-k$-tuple */
  end
end
return *The center of the cell with maximum votes and its corresponding set of intersecting surfaces.*

that decomposes the search space to significantly improve the runtime complexity. We round surfaces so that we can group similar surfaces for joint processing, without affecting the outcome of the computation. This canonization means that in every recursive step of the algorithm (similar to levels in an octree decomposition) there are approximately *the same number of surfaces* to process. The consequence is a worst case runtime complexity of $O(n + {}^{\text{polylog}(1/\varepsilon)}/_{\varepsilon^{\ell+d-k}})$ (see Theorem 4.4 in [1]) or $O(n + {}^{1}/_{\varepsilon^{\ell+d-k}})$ ignoring log factors. For sufficiently large n generalized voting is thus always asymptotically faster than naive voting because of the multiplicative influence of n on the approximation cost in naive voting.

Algorithm 2: Efficient Generalized Voting

Data: S: surfaces, B: box in $[0,1]^d$, ε: distance
Result: Point in B (among all ε-grid points) with the maximum of ε-close surfaces.
Function `SurfaceConsensus` (S, B, ε) **:**
  if $Diam(B) \leq \varepsilon$ **then**
    return (B_c, S) /* B_c is the center of B */
  end
  Canonize all surfaces S (for B) to a new set S_c.
  Subdivide B to 2^d sub-boxes, B_i.
  For each sub-box B_i, find a subset $S_{c_i} \subset S_c$ of surfaces that intersect it.
  for *all* S_{c_i} **do**
    (p_i, I_i) = `SurfaceConsensus` $(S_{c_i}, B_i, \varepsilon)$
    /* p_i: a point in B_i, I_i: the set of inliers */
  end
  return $(p_k = \text{argmax}_{p_i} |I_i|, I_k)$

3.2.1 Algorithm and Implementation

Given an outlier removal problem, the first step is to formulate surfaces with a parametric representation as in Eq. (1).

From constraints to general voting. We start with a set of constraints, each of which implicitly defines a surface embedded in R^d (by the points that satisfy the constraint). The surface may have only $k \leq d$ dimensions, meaning that given $x_1, ..., x_k$, we can compute the other $x_{k+1}, ...x_d$. To use our framework, we have to provide two functions:

1. A predicate that returns whether a given surface intersects a box in R^d.
2. A function $F^{(\sigma)}(\mathbf{x}, \mathbf{t}) + \mathbf{f}$ that computes the $d-k$ dependent variables, given the other k variables.

In the example of 2D line fitting, Eq. (2), every 2D point (x_1, x_2) defines a line in voting space and with $k = 1$, we have a $d - k = 1$ dimensional surface embedded in the 2D ambient space. Function (1) computes whether $ax_1 + b$ intersects the given box B and function (2) is $x_2 = ax_1 + b$.

Canonization. Most Branch&Bound algorithms subdivide the parameter space, discarding branches early based on computed scores. In 3D, a common such subdivision is the octree; in the following we will call the subdivision scheme an octree, independent of the actual dimensionality. A key ingredient that makes our generalized voting algorithm efficient is *Canonization*; a surface rounding process we apply before recursing to the next level of the octree. Surfaces which are close to each other in the current box are rounded and grouped into the same surface, thus bounding the overall number of surfaces.

Each surface $\sigma_{\mathbf{t},\mathbf{f}}$ is rounded carefully to $\sigma_{\mathbf{s},\mathbf{g}}$ to not change intersections on the finest level of the octree:

$$|(F_j(\mathbf{x};\mathbf{t}) + \mathbf{f_j}) - (F_j(\mathbf{x};\mathbf{s}) + \mathbf{g_j})| \leq \varepsilon \quad j = k+1, ..., d$$

where ε is the "target closeness" introduced in Section 2. Due to this bound, the surface rounding does not change the optimization result, while significantly reducing the number of surfaces to process. During canonization we keep track of merged surfaces to recover the original surfaces (the set of inliers) after finding the maximum.

Due to the depth of the octree, we round each surface at most $\log 1/\varepsilon$ times, each time introducing a rounding error c that is defined by the Lipschitz constant, a bound on the surface gradient. So the error we can tolerate in one rounding step is $\varepsilon' = \frac{\varepsilon}{c \log 1/\varepsilon}$. Using this constant, [1] derives rounding rules for the surface parameters: Round free parameters to multiples of ${}^{\varepsilon'}/_{(\ell+1)}$ and essential parameters to multiples of ${}^{\varepsilon'}/_{(\ell+1)\delta}$ where δ is the diameter of the current box of the octree. See [1] for additional detail.

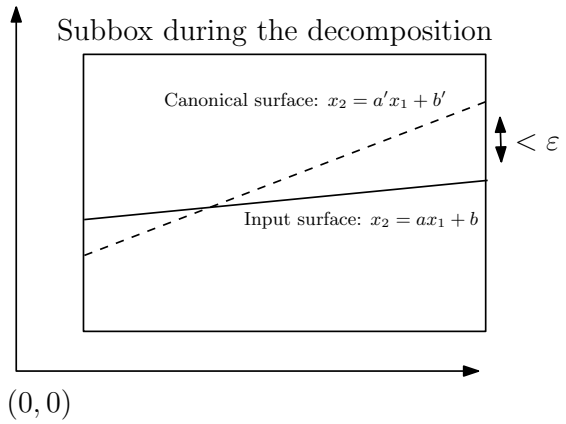

Figure 1: Canonization where the surface is a line in 2D, which is intersected with a 2D box in the search quadtree.

Figure 1 shows the rounding process for the case of a 2D surface given by $x_2 = ax_1 + b$ and a box B. Note that our goal is to find a surface which is close to the input surface *within the box* B. We first round the essential parameter a to a' which moves the line away from the box, so we translate it by changing b and then rounding it to b'. The key result is that the number of canonical surfaces in B is upper bounded independent of the number of input surfaces.

In practice, ε' determines the runtime upper bound and the best parameter can vary among coordinates such that we tune constants per parameter (e.g. different ε' for a and b in the line fitting example).

General surface-box intersection algorithm for any surface and box: Suppose the surface σ is given as a polynomial equation $F(x_1, x_2, x_3) = 0$ and that the box is $B = [0, 1]^3$. If we assume σ is connected, then σ intersects B if either (1) σ is fully contained in B, or (2) it intersects some face, or (3) it intersects some edge. To test for case (1), pick an arbitrary point on σ and test if it lies in B. If σ is not connected, repeat this for one point in each connected component. Cases (2) and (3) are handled recursively: for (2), intersect each face with the 2D plane, e.g. with $F(x_1, x_2, 0) = 0$, by checking a point on (each connected component of) the surface and each edge of the face. For (3) we have a univariate polynomial, e.g. $F(x_1, 0, 0) = 0$, and we need to test if it has a root in $[0, 1]$.

This intersection algorithm can become slow for many dimensions but we found that using a subset of the conditions is a good approximation in practice.

3.3. Theoretical Analysis of Related Work

To the best of our knowledge three alternatives to our algorithm exist, for which we compare asymptotic runtime.

RANSAC [8] is the most common approach to solve outlier rejection problems. The RANSAC complexity is $O\left(\frac{n}{\log(1-b^k)}\right)$ for a lower bound b on the fraction of inliers, a minimal set size k needed to define a possible solution and n input constraints. This is linear in n only when b is a constant and even then it grows quickly when b is decreasing, making RANSAC unattractive for large scale problems.

Hough based methods [11, 7] exist in many variants, but all share a term with polynomial runtime complexity in which all sets of points (each of minimal size k) are traversed. For each subset we vote for the parameters defined by the subset, taking $O(n^k)$ time to find the parameters with the maximum number of votes. Alternatively for each single point p one can enumerate all bins in the Hough space corresponding to parameters of models passing through (or close) to p. This takes $O(n/\varepsilon^s)$ where s is the dimension of the Hough surface. A randomized version of Hough voting is applicable, if a lower bound on the number of inliers exists, though, it introduces the same limitation as RANSAC.

Branch&Bound [14] methods have been considered and implemented for a large number of optimization problems, including the family we consider here [3, 16, 5, 9, 10]. They are optimal up to an error bound defined by the smallest box that terminates the process (say of size ε assuming w.l.o.g. that the space is $[0, 1]^d$). The practical runtime of Branch&Bound can be low but is highly dependent on the structure of the surfaces and the quality of the bounds which are notoriously hard to find. The worst case runtime is $O(n/\varepsilon^k)$ (ignoring the log factor), where k depends on the dimension of the problem, and is thus similar to the naive enumeration of cells. Recently, the 6DOF posing problem was solved without correspondences using B&B [6], against which we evaluate in Section 5. The worst case of [6] is $O(\mu^{-3}\eta^{-6}nm)$ for m 3d points, n 2d points and μ, η are tolerance parameters similar to our ε. This aligns with the naive method, as nm is the number of matches.

Our generalized voting approach is asymptotically and in practice faster and more general than these alternatives while being easy to implement.

4. Applications

In the following, we enumerate a list of typical outlier rejection problems in geometric computer vision and we show how each one of them can be reduced to the incidence problem and solved efficiently using our method. We deliberately do not compare to the numerous state of the art methods across all applications, but rather want to emphasize the *generality of our approach*.

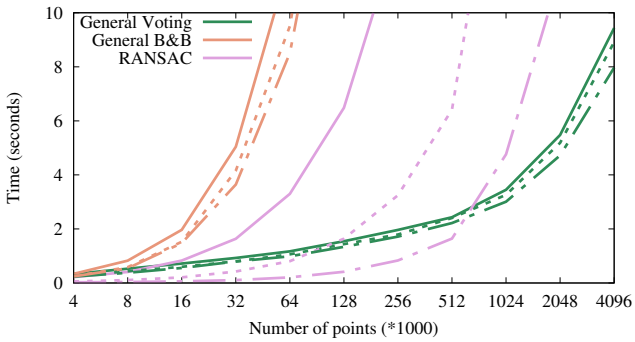

Figure 2: Runtime comparison for line fitting with various inlier fractions: solid 1%, dotted 2%, dashed 4%.

4.1. Fitting Hyperplanes in d-space

Model fitting is a well investigated problem, where the goal is to report the hyperplane ε-close to most points. It is commonly solved using RANSAC or Hough voting [20, 7, 11]. It is also solved using a primal-dual method in [1].

In order to apply our general method, we first use the point-hyperplane duality [2]. Points are transformed to hyperplanes and vice versa, preserving the vertical distance between points and planes. If we keep hyperplanes in appropriate orientation this algebraic distance (the d coordinate) is a good approximation to the Euclidean distance we want to minimize. We then search for the point that is close to the maximum number of hyperplanes and transform it back via duality to obtain the best fitting hyperplane. For example the parameterization in 3D is the standard plane equation:

$$x_d = a_0 + \Sigma_{i=1}^{d-1} x_i a_i. \tag{3}$$

with $\ell = d-1, k = d-1$ in R^d and the runtime is $O(n + \frac{\text{polylog}(1/\varepsilon)}{\varepsilon^d})$ compared to $O(n/\varepsilon^{d-1})$ in the naive method. The d-dimensional algorithm is then better for any $n > 1/\varepsilon$ (for any fixed d).

Evaluation We evaluate the runtime for the 2D case by comparing B&B and Ransac to our method where we implemented the problem specific surface definition and the intersection predicate. "General B&B" refers to our general method with all its ingredients except canonization. The test data consists of uniformly sampled outlier points in the unit cube and inlier points sampled along a fixed line with additive Gaussian noise with standard deviation 0.1. We experiment with an increasing number of points and different inlier fractions, which we provide to RANSAC. Fig. 2 shows how RANSAC's runtime strongly depends on the inlier fraction and our method compares preferably as the number of points grows. B&B is slowest, since the uniform outliers hit the worst case of the algorithm. In all cases, accurate line parameters were found using $\varepsilon = 0.002$.

4.2. Absolute Camera Posing Problems

Related work [23, 21, 1] defines the surfaces for posing calibrated cameras with known gravity direction, resulting in a 4 DoF problem. Here, we derive surfaces for higher DoF that are difficult to handle by naive voting due to the dimensionality. Let $\mathbf{w} = (w_1, w_2, w_3)$ be a point in $\mathbb{R}^3$ and (ξ, η) a point in the normalized image plane. The triple $(\mathbf{w}, \xi, \eta)$ is a correspondence $\mathbf{c}$ and we show its transformation into a surface $\sigma_{\mathbf{c}}$. See Table 1 for runtime complexities.

4.2.1 Unknown focal length, but known gravity direction (5DoF)

To solve this problem we search for $(x, y, z, \kappa, f) \in \mathbb{R}^5$ where (x, y, z) is the camera position, $\kappa = \tan\theta$ with θ denoting the remaining camera orientation around gravity, and f is the focal length. Each such point, models a possible pose and focal length of the camera. A correspondence $\mathbf{c}$ is parameterized by the triple $(\mathbf{w}, \xi, \eta)$ and defines a 3-dimensional algebraic surface $\sigma_{\mathbf{c}}$. It is the surface of all camera poses and focal lengths that see $\mathbf{w}$ at image coordinates $(\xi f, \eta f)$. We can derive the following parametric representation of $\sigma_{\mathbf{c}}$, expressing z and κ as functions of x,y and f (Please see [1, Sec. 2.2] for the derivation).

$$\kappa = \frac{(w_2 - y) - \xi f (w_1 - x)}{(w_1 - x) + \xi f (w_2 - y)} \tag{4}$$

$$z = w_3 - \eta f \sqrt{(w_1 - x)^2 + (w_2 - y)^2} \tag{5}$$

Our goal is to find the point (x, y, z, κ, f) that is ε-close to the maximal number of surfaces. In other words the ray from the camera center c to $\mathbf{w}$ goes approximately (define by ε) through $(\xi f, \eta f)$ in the image plane.

In the case of our 3-surfaces in 5-space, the parameter w_3 is free, and we introduce a second artificial free parameter into equation 4 for κ. We have $\ell = 4$ essential parameters: w_1,w_2,ξ, and η. With $d = 5$ and $k = 3$, Algorithm 2 obtains worst case complexity $O(n + \frac{\text{polylog}(1/\varepsilon)}{\varepsilon^6})$. Naive voting takes $O(n/\varepsilon^3)$ time so that the general technique is asymptotically better for any $n > 1/\varepsilon^3$ (ignoring poly logarithmic factors).

4.2.2 Unknown focal length (7DoF)

We use $(x, y, z, \boldsymbol{\phi}, f)$ as the 7-tuple of unknowns that we aim to solve for, where $\boldsymbol{\phi}$ is a 3-vector describing the minimal rotation parameterization (e.g. angle-axis). Given the standard general 3×4 projection matrix $P = \text{diag}(f, f, 1) \begin{bmatrix} R(\boldsymbol{\phi}) & \mathbf{t} \end{bmatrix}$ the world point $\mathbf{w}$ projects to

$$\lambda \begin{pmatrix} \xi \\ \eta \\ 1 \end{pmatrix} = P \begin{pmatrix} \mathbf{w} \\ 1 \end{pmatrix} = \begin{pmatrix} f(\mathbf{r_1 w} + x) \\ f(\mathbf{r_2 w} + y) \\ \mathbf{r_3 w} + z \end{pmatrix}, \tag{6}$$

Posing problem	Known focal	Known gravity	Generalized voting	Naive voting	General faster than naive for
4DoF [1]	✓	✓	$\tilde{O}\left(n+1/\varepsilon^6\right)$	$O(n/\varepsilon^2)$	$n > 1/\varepsilon^4$
5DoF	-	✓	$\tilde{O}\left(n+1/\varepsilon^6\right)$	$O(n/\varepsilon^3)$	$n > 1/\varepsilon^3$
6DoF	✓	-	$\tilde{O}\left(n+1/\varepsilon^7\right)$	$O(n/\varepsilon^4)$	$n > 1/\varepsilon^3$
7DoF	-	-	$\tilde{O}\left(n+1/\varepsilon^7\right)$	$O(n/\varepsilon^5)$	$n > 1/\varepsilon^2$
5DoF (radial camera)	-	-	$\tilde{O}\left(n+1/\varepsilon^6\right)$	$O(n/\varepsilon^4)$	$n > 1/\varepsilon^2$

Table 1: Complexity analysis for the camera posing formulations of Sec. 4.2. $\tilde{O}(n+1/f(\varepsilon)) := O(n+\text{polylog}(1/\varepsilon)/f(\varepsilon))$.

where $\mathbf{r_i}$ denote the i^{th} row of the rotation matrix. This reveals constraints

$$\begin{aligned} 0 &= f(\mathbf{r_1 w} + x) - \xi(\mathbf{r_3 w} + z) \\ 0 &= f(\mathbf{r_2 w} + y) - \eta(\mathbf{r_3 w} + z) \end{aligned} \tag{7}$$

which allows to parameterize x and y as a a function of $f, \boldsymbol{\phi}, z$ for each correspondence $(\mathbf{w}, \xi, \eta)$ according to

$$\begin{aligned} x &= \xi(\mathbf{r_3 w} + z)/f - \mathbf{r_1 w} \\ y &= \eta(\mathbf{r_3 w} + z)/f - \mathbf{r_2 w}. \end{aligned} \tag{8}$$

With $d = 7, \ell = 5, k = 5$ the naive rendering takes $O(n/\varepsilon^5)$ while general voting takes $O(n + \frac{\text{polylog}(1/\varepsilon)}{\varepsilon^7})$ which is better for any $n > 1/\varepsilon^2$.

4.2.3 Calibrated camera (6DoF)

This is identical to the 7 DoF case where we set $f = 1$. With $d = 6, \ell = 5, k = 4$, naive rendering takes $O(n/\varepsilon^4)$ while generalized voting takes $O(n + \frac{\text{polylog}(1/\varepsilon)}{\varepsilon^7})$ as for 7 DoF case which is better for any $n > 1/\varepsilon^3$.

4.2.4 Unknown focal length using a radial camera model (5DoF)

The previous formulation for the 6DoF pose plus focal length case requires a 7-dimensional voting space. As an alternative we propose to leverage a radial camera model [22] which is known to work well for pose estimation with unknown focal length [4, 15]. The approach factors the pose estimation in two consecutive steps where the first relies on line-to-point correspondences and solves for all parameters except the focal length and the camera motion along the optical axis. A second *upgrade* step solves for the remaining parameters using a least squares fit. For outlier removal we focus on the first step and therefore are looking for a 5-tuple $(x, y, \boldsymbol{\phi})$. The 2×4 radial camera projection matrix is

$$P = \begin{bmatrix} \mathbf{r_1} & -y \\ \mathbf{r_2} & x \end{bmatrix} \tag{9}$$

and we have $[\mathbf{w}^T, 1](\eta \mathbf{p_1} - \xi \mathbf{p_2}) = 0$, where $\mathbf{p_1}, \mathbf{p_2}$ are the rows of P. We then obtain $\sigma_{\mathbf{w}}$ where we parameterize x as a function of $y, \boldsymbol{\phi}$:

$$x = \eta(\mathbf{r_1 w} - y)/\xi - \mathbf{r_2 w}. \tag{10}$$

We have $d = 5, \ell = 5, k = 4$ and the algorithm takes $O(n + \frac{\text{polylog}(1/\varepsilon)}{\varepsilon^6})$ time, compared to $O(n/\varepsilon^4)$ with naive voting, the general algorithm being faster for any $n > 1/\varepsilon^2$.

4.2.5 Evaluation

Localizing an uncalibrated camera with a known axis of rotation is a common problem in computer vision. Both the problem dimensionality and the number of unknowns are well suited to demonstrate the general use of our method. Therefore, we focus on the 5 DoF problem described in Section 4.2.1 here and compare the runtimes of our generalized voting approach to naive voting, Branch&Bound, and RANSAC.

Our evaluation data exhibits real-world, large-scale urban scenes where a set of 3D points and potential image correspondences are given. We match real query images against 3D points of a large SfM model. Using 7, 14, and 56 candidates in the nearest neighbor search results in problem sizes of roughly $10k$, $15k$, and $20k$ matches.

For the solution computations we consider typical space limitations for all methods: Due to the nature of the parameterization we only consider a camera orientation with tangent in $[-1, 1]$, rotating the scene accordingly if needed. We bound the spatial position to be at most 50m and the camera height 5m from the ground truth. The focal length is bounded within a fraction $[0.6, 1.3]$ of the ground truth.

Figure 3 illustrates that our method is most efficient. In order to eliminate implementation details which can change runtimes considerably, we do not measure time but instead count the number of dominant operations. That is the number of 5D grid cells that are touched in the naive implementation and the number of calls to the surface-box intersection predicate for our algorithm and for B&B. For RANSAC we simplified the problem to one with known focal length and were thus able to use a 3-point minimal solver with early rejection to account for the known gravity direction. We also tuned the number of iterations and inlier tolerance to find a good pose with minimal runtime. As expected, the gap between the methods and the improvement in the proposed algorithm is increasing with the size of the input as suggested by the asymptotic complexity.

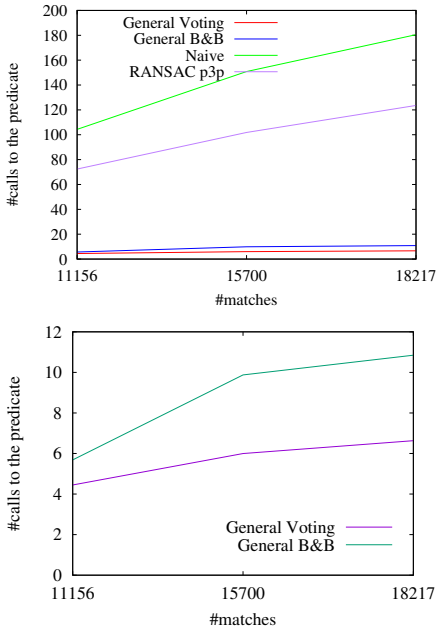

Figure 3: Runtime comparison for 5 DoF camera posing: (top) high cost of naive voting; (bottom) zoom-in on B&B and our generalized voting.

4.3. Intersections of Rays/Lines in 3-space

The problem consists of a set of n rays in 3-space and the grid cell size ε. The task is to report all subsets of rays intersecting any cell and with cardinality larger than some threshold. We can formulate the general surfaces using:

$$y = ax + b \qquad z = cx + d. \tag{11}$$

With $\ell = 2, k = 1, d = 3$ the naive algorithm takes $O(n/\varepsilon)$ while the general voting takes $O(n + \frac{\text{polylog}(1/\varepsilon)}{\varepsilon^4})$ which is better for any $n > 1/\varepsilon^3$.

4.4. Pointset Alignment

In this problem, A and B are sets of points in the plane. Any hypothetical correspondence between $p \in A$ and $q \in B$ defines a surface in the 4-space of similarity transformations (translation, rotation and scale). The linear parametrization is:

$$q_x = ap_x + bp_y + c \tag{12}$$

$$q_y = -bp_x + ap_y + d \tag{13}$$

where $a = s\cos(\theta), b = s\sin(\theta), s$ is the scale, θ is the rotation and (c, d) is the translation vector. The general surface is then:

$$a = -\frac{cp_x + dp_y - p_xq_x - p_yq_y}{p_x^2 + p_y^2} \tag{14}$$

$$b = -\frac{cp_y - dp_x - p_yq_x + p_xq_y}{p_x^2 + p_y^2}. \tag{15}$$

Which is a 2-surface embedded in R^4 where a, b are given as a functions of c, d. In this case we have two essential parameters (c, d) and we introduce two artificial new free parameters. With $\ell = 2, k = 2$ and $d = 4$ with n as the number of canonical surfaces the naive algorithm would take $O(\frac{n}{\varepsilon^2})$ and the general voting takes $O(n + \frac{1}{\varepsilon^4}\log\frac{1}{\varepsilon})$ which is better for any $n > \frac{1}{\varepsilon^2}$ (ignoring log factors).

5. Comparison with GOPAC [6]

Solving the 6 DoF camera posing problem without known correspondences is hard due to the large search space. Recently, [6] introduced a solution using a globally optimal method based on Branch&Bound and conducted extensive evaluations on public datasets (Data61/2D3D [18] and Stanford 2D-3D-Semantics (2D-3D-S) [13]) against state-of-the-art alternatives.

We evaluate our general voting on this problem, viewing it as a Branch&Bound search. We compare our results with [6] on the same datasets, using the GOPAC code provided by the authors (re-run on our machine to ensure comparability to our approach), showing that our general algorithm is not only asymptotically better but also faster in practice. Both our algorithm and GOPAC, are globally optimal up to a prescribed resolution ε and perform joint inlier set maximization and correspondence search. Our method achieves a significantly better worst case runtime of $O(n + \text{poly}(1/\varepsilon))$ compared to $O(n\text{poly}(1/\varepsilon))$ in GOPAC.

In order to implement the 6DOF solver in our general framework we reformulated the problem as a surface consensus problem according to Sec. 4.2.3. Compared to GOPAC our formulation does not use an angular error metric, but is based on surface distances. However, it is possible to obtain the globally optimal solution by conservative expansion of the cubes (to avoid missing inliers) and verify the angular projection error on the final inlier set for the minimal cuboids.

Data61/2D3D [18] outdoor dataset: The dataset consists of a 88 3D points and 11 sets of 30 bearing vectors. Table 2 shows the localization performance and runtime for GOPAC, our generalized voting and RANSAC. Both Branch&Bound algorithms show comparable accuracy due to their global optimality and we only expect a difference in runtime. Here our algorithm is more than an order of magnitude faster. Equivalent to [6] we restrict the translation solution to a $50m \times 5m \times 5m$ domain along the street (as it is known that cameras are mounted on a vehicle), and a camera is considered successfully posed, if the rotation error is less than 0.1 radians and the normalized translation error is less than 0.1. There is no prior knowledge here, we have used limitations done in [6] for a fair comparison.

Algorithm	GOPAC	GV (ours)	RANSAC
Translation Error (m)	2.76	2.89	28.5
Rotation Error (deg)	2.18	0.46	179
Runtime (s)	457	27	422
Success rate (inliers)	1	1	0
Success rate (pose)	0.82	0.82	0.09

Table 2: Camera posing results (median error over the 11 queries) for Scene 1 of the Data61/2D3D dataset. Error metrics for GOPAC are taken from [6]). Runtimes are from single-threaded execution of C++ code, where GOPAC was rerun on our machine.

Algorithm	GOPAC	GV (ours)
Translation Error (m)	0.29	0.38
Rotation Error (deg)	3.46	2.81
Runtime (s)	508	421
Success rate (inliers)	1	1
Success rate (pose)	0.77	0.77
Success rate within 60s	0.19	0.42

Table 3: Camera posing results (median error) for Area 3 of the Stanford 2D-3D-S dataset (see Tab. 3 in [6]). Runtimes are from running C++ code on CPU with 8-threads.

Stanford 2D-3D-Semantics [13] indoor dataset: The dataset consists of 15 rooms of different types and 27 sets of 50 bearing vectors. Table 3 lists our performance and runtime comparison. Since RANSAC has polynomial runtime in the inlier ratio, while our approach is independent of it, we forwent more RANSAC comparisons. As the experimental evaluation of [6] uses a GPU implementation we reran both algorithms on CPU using 8 threads to ensure comparability. In order to obtain reasonable runtimes we set ε to 0.5m and 0.1 radians, respectively for both methods. Again we expect a similar localization performance due to the global optimality of both methods. On average the runtime of our algorithm on this dataset is only slightly better. As noted in Section 3.2.1, the effect of the asymptotically better worst case runtime increases with the hardness of the problem (both, in size and data distribution). Therefore, Figure 4 depicts the (sorted) runtimes for all queries, which illustrates that our algorithm becomes significantly faster for the hard cases.

6. Conclusion

In this paper we introduced the concept of *general voting*, a powerful method for outlier rejection applicable to a wide range of geometric computer vision problems. We adopt the previously proposed method of *approximate incidences* [1] to solve for inlier maximization in multiple classical computer vision problems ranging from camera posing and ray intersection to geometric model fitting. We described the general recipe with a simple to understand 2d-line fitting example, but also demonstrated its applicability to real-world problems.

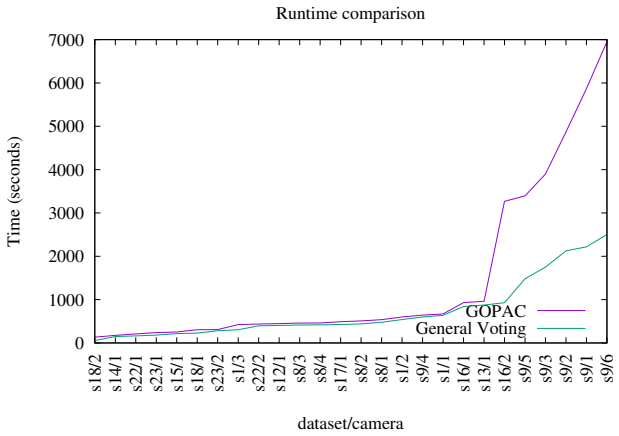

Figure 4: Sorted runtimes for all queries in the Stanford 2D-3D-S dataset. Same experiment as Table 3.

Through theoretical analysis and experiments we demonstrated that our algorithm scales better than its alternatives, like RANSAC or Branch&Bound, both in terms of complexity and real-world runtime. The experimental data validated that our solution performs particularly well for large problems with low inlier ratios where alternatives require problem specific knowledge to remain applicable.

One of the key use-cases we investigated is camera posing with and without known gravity direction and focal length, problems that cannot be solved efficiently at large scale with previously published methods, yet that have wide applicability in industry. To solve these cases, we introduced two algorithms that are key to efficiency: canonization of the intersected surfaces and an efficient d-box intersection algorithm which we combine in a spatial subdivision scheme. To demonstrate the impact of these contributions we provided an extensive evaluation against a recently published state-of-the art method on publicly available, large-scale indoor and outdoor datasets.

Beside solving concrete localization approaches this work introduced the concept of general voting to the wider computer-vision community. We aim to provide a recipe for applying this approach to a range of problems and publish an open-source implementation of our efficient general voting to unlock new applications and research directions.

Acknowledgements The authors would like to thank Micha Sharir for helpful discussions concerning the general surface-box intersection.

References

[1] Dror Aiger, Haim Kaplan, Efi Kokiopoulou, Micha Sharir, and Bernhard Zeisl. General techniques for approximate incidences and their application to the camera posing problem. In *Symposium on Computational Geometry*, pages 8:1–8:14, 2019. 1, 2, 3, 5, 6, 8

[2] E. Artin. 1.4 duality and pairing. *Geometric Algebra*, 1957. 2, 5

[3] T. M. Breuel. Implementation techniques for geometric branchand-bound matching methods. *Comput. Vis. Image Understanding*, 90(3):258–294, 2003. 1, 4

[4] Martin Bujnak, Zuzana Kukelova, and Tomás Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. *ACCV*, 1:11–24, 1010. 6

[5] F. Kahl C. Olsson and M. Oskarsson. Branch-and-bound methods for euclidean registration problems. *PAMI*, 31(5):783–794, 2009. 4

[6] Dylan Campbell, Lars Petersson, Laurent Kneip, and Hongdong Li. Globally-optimal inlier set maximisation for camera pose and correspondence estimation. *PAMI*, 42(2):328–342, 2020. 1, 2, 4, 7, 8

[7] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. ACM*, 15:11–15, 1972. 4, 5

[8] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1, 4

[9] Johan Fredriksson, Viktor Larsson, Carl Olsson, Olof Enqvist, and Fredrik Kahl. Efficient algorithms for robust estimation of relative translation. *Image Vis. Comput.*, 52:114–124, 2016. 2, 4

[10] Johan Fredriksson, Viktor Larsson, Carl Olsson, and Fredrik Kahl. Optimal relative pose with unknown correspondences. *CVPR*, 52:1728–1736, 2016. 2, 4

[11] P.V.C. Hough. Method and means for recognizing complex patterns. *U.S. Patent 3,069,654*, 1962. 1, 4, 5

[12] D. P. Huttenlocher and K. Kedem. Computing the minimum hausdorff distance for point sets under translation. In *Symposium on Computational Geometry*, pages 340–349, 1990. 2

[13] A. R. Zamir I. Armeni, A. Sax and S. Savarese. Joint 2d-3dsemantic data for indoor scene understanding. In *ArXiv e-prints, Feb. 2017. [Online]. Available: http://adsabs.harvard.edu/abs/2017arXiv170201105A*, 2017. 7, 8

[14] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: J. Econometric Soc.*, 28:497–520, 1960. 1, 4

[15] Viktor Larsson, Torsten Sattler, Zuzana Kukelova, and Marc Pollefeys. Revisiting radial distortion absolute pose. *ICCV*, 1:1062–1071, 2019. 6

[16] H. Li and R. Hartley. The 3d-3d registration problem revisited. *Proc. IEEE Int. Conf. Comput. Vis*, pages 1–8, 2007. 4

[17] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *European conference on computer vision*, pages 15–29. Springer, 2012. 2

[18] M. Salzmann S. T. Namin, M. Najafi and L. Petersson. A multimodal graphical model for scene analysis. In *Winter Conf. Appl. Comput. Vis.*, page 1006–1013, 2015. 7

[19] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8601–8610, 2018. 2

[20] Linda Shapiro and George Stockman. Computer vision. *Prentice-Hall, Inc.*, 2001. 5

[21] Linus Svärm, Olof Enqvist, Fredrik Kahl, and Magnus Oskarsson. City-scale localization for cameras with known vertical direction. *PAMI*, 39(7):1455–1461, 2016. 1, 5

[22] SriRam Thirthala and Marc Pollefeys. Radial multi-focal tensors - applications to omnidirectional camera calibration. *IJCV*, 96(2):195–211, 2012. 6

[23] Bernhard Zeisl, Torsten Sattler, and Marc Pollefeys. Camera pose voting for large-scale image-based localization. In *ICCV*, pages 2704–2712, 2015. 1, 5