

APPENDIX

A. Rotation Uncertainty Angle: Implementation Details

Lemma 3, copied below, requires the evaluation of $\max_{\mathbf{r} \in \mathcal{S}_r} \angle(\mathbf{R}_r \mathbf{p}, \mathbf{R}_{\mathbf{r}_0} \mathbf{p})$, where \mathbf{p} is any 3D point, \mathbf{r}_0 is the angle-axis vector at the centre of rotation cube \mathcal{C}_r with surface \mathcal{S}_r and \mathbf{R}_r is the rotation matrix induced by angle-axis vector \mathbf{r} .

Lemma 3. (*Rotation uncertainty angle*) Given a 3D point \mathbf{p} and a rotation cube \mathcal{C}_r centred at \mathbf{r}_0 with surface \mathcal{S}_r , then $\forall \mathbf{r} \in \mathcal{C}_r$,

$$\angle(\mathbf{R}_r \mathbf{p}, \mathbf{R}_{\mathbf{r}_0} \mathbf{p}) \leq \min(\max_{\mathbf{r} \in \mathcal{S}_r} \angle(\mathbf{R}_r \mathbf{p}, \mathbf{R}_{\mathbf{r}_0} \mathbf{p}), \pi) \triangleq \psi_r(\mathbf{p}, \mathcal{C}_r). \quad (1)$$

Proof. Inequality (1) can be derived as follows:

$$\angle(\mathbf{R}_r \mathbf{p}, \mathbf{R}_{\mathbf{r}_0} \mathbf{p}) \leq \min(\max_{\mathbf{r} \in \mathcal{C}_r} \angle(\mathbf{R}_r \mathbf{p}, \mathbf{R}_{\mathbf{r}_0} \mathbf{p}), \pi) \quad (2)$$

$$= \min(\max_{\mathbf{r} \in \mathcal{S}_r} \angle(\mathbf{R}_r \mathbf{p}, \mathbf{R}_{\mathbf{r}_0} \mathbf{p}), \pi) \quad (3)$$

where (3) is a consequence of the order-preserving mapping, with respect to the radial angle, from the convex cube of angle-axis vectors to the spherical surface patch, since the mapping is obtained by projecting from the centre of the sphere to the surface of the sphere. \square

While it is possible to calculate the bound by sampling the cube surface using a grid of step-size σ_g , evaluating the angle at each sample and adding $\sqrt{2}/2 \times \sigma_g$ to the greatest angle calculated (by Lemma 1), it is significantly more computationally efficient to use a different approach. We make two assumptions: (i) the maximum always occurs on the cube skeleton (edges and vertices), not the faces; and (ii) the angle function along each edge is quasiconvex or concave (specifically unimodal). The former assumption (i) has been demonstrated empirically in simulations and can be seen in Figure A.1, where it can be observed that rotation vectors on the cube faces are not projected beyond the convex hull of the projection of the edges for a given point. Therefore, the projected angle maximiser can always be found on an edge or vertex. The latter assumption (ii) has also been demonstrated empirically in simulations for all rotation cubes used in the GOPAC algorithm (that is, octree subdivisions of the angle-axis cube $[-\pi, \pi]^3$). In the vast majority of cases, the function is (quasi)convex, meaning that the angle maximiser occurs at one of the two vertices joined by the edge (the extreme points). In a small fraction of cases, the maximum occurs on the edge, but not at a vertex, as in Figure A.1. In these cases the assumption of unimodality allows us to use an efficient search routine, golden section search, which does not require the time-consuming evaluation of the derivative.

Our approach for calculating the rotation uncertainty angle $\psi_r(\mathbf{f}, \mathcal{C}_r)$ for a bearing vector \mathbf{f} and rotation cube \mathcal{C}_r , centred at angle-axis vector \mathbf{r}_0 with vertices $\{\mathbf{r}_i\}_{i \in [1,8]}$ and an edge parametrisation of $\mathbf{r}(\lambda) = \mathbf{r}_i + \lambda(\mathbf{r}_j - \mathbf{r}_i)$, is as follows.

- (i) For each edge, evaluate the sign of the derivative of the angle function $\angle(\mathbf{R}_{\mathbf{r}(\lambda)}^{-1} \mathbf{f}, \mathbf{R}_{\mathbf{r}_0}^{-1} \mathbf{f})$ with respect to λ at $\lambda = 0$ and $\lambda = 1$ using (12) and (13);
- (ii) if (12) is positive and (13) is negative, use golden-section search [5] to find the angle maximiser on that edge, using a tolerance of $\pi/2048$, and add $\pi/2048$ to the result;
- (iii) otherwise, the angle maximiser on that edge is a vertex: evaluate the angle with respect to the projected cube centre at both vertices and choose the maximum; and
- (iv) choose the overall maximum angle as ψ_r .

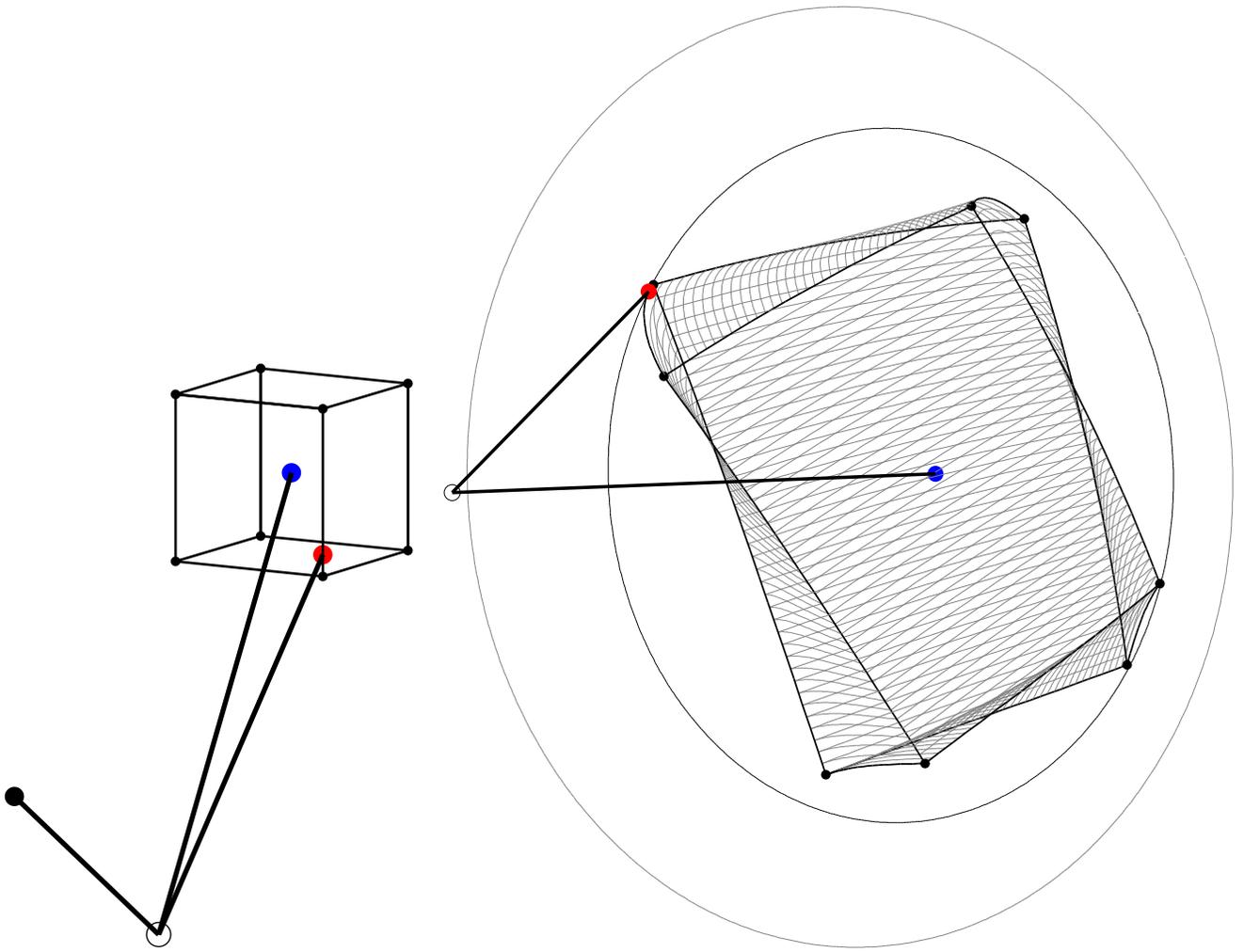
Note that golden section search terminates at a tolerance of $\pi/2048$. By Lemma 1, the bound is therefore at most $\pi/2048 = 0.088^\circ$ incorrect, a value that is added to the upper bound to ensure correctness.

We now provide the derivative of the rotation angle function in Lemma A.1. Given a unit 3D bearing vector \mathbf{f} and a rotation cube \mathcal{C}_r centred at \mathbf{r}_0 with vertices $\{\mathbf{r}_i\}_{i \in [1,8]}$, the rotation angle function with respect to λ , for an edge parametrisation of $\mathbf{r}(\lambda) = \mathbf{r}_i + \lambda(\mathbf{r}_j - \mathbf{r}_i)$ with $\lambda \in [0, 1]$, is

$$A(\lambda) = \arccos((\mathbf{R}_{\mathbf{r}_0}^{-1} \mathbf{f}) \cdot (\mathbf{R}_{\mathbf{r}(\lambda)}^{-1} \mathbf{f})). \quad (4)$$

Lemma A.1. (*Derivative of the rotation angle function*) Given a unit 3D bearing vector \mathbf{f} and a rotation cube \mathcal{C}_r centred at \mathbf{r}_0 with vertices $\{\mathbf{r}_i\}_{i \in [1,8]}$, then

$$\frac{dA}{d\lambda} = \frac{-\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}(\lambda)}^\top [\mathbf{f}]_\times \left(\mathbf{r}(\lambda) \mathbf{r}(\lambda)^\top - (\mathbf{R}_{\mathbf{r}(\lambda)} - I) [\mathbf{r}(\lambda)]_\times \right) (\mathbf{r}_j - \mathbf{r}_i)}{\|\mathbf{r}(\lambda)\|^2 \sqrt{1 - (\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}(\lambda)}^\top \mathbf{f})^2}}. \quad (5)$$



(a) Rotation cube in angle-axis space with centre \mathbf{r}_0 (blue dot), projected angle maximiser \mathbf{r}^* (red dot), $\mathbf{R}_{\mathbf{r}_0} \mathbf{p}$ (blue dot), angle maximiser $\mathbf{R}_{\mathbf{r}^*} \mathbf{p}$ (red dot) and origin (black circle). Cube edges and vertices are shown as thin black lines and small black dots respectively. (b) Rotation of 3D point \mathbf{p} by angle-axis vectors on the surface of the cube with centre-rotated point $\mathbf{R}_{\mathbf{r}_0} \mathbf{p}$ (blue dot), angle maximiser $\mathbf{R}_{\mathbf{r}^*} \mathbf{p}$ (red dot) and origin (black circle). 40 equally-spaced lines across each face are plotted in grey. All points and lines, other than the origin and lines to the origin, lie on the surface of a sphere with radius $\|\mathbf{p}\|$. Cube edges and vertices are shown as thin black lines and small black dots respectively. The weak rotation uncertainty angle ψ_r^w corresponds to the aperture angle of the cone formed by the origin and the grey circle. Our rotation uncertainty angle ψ_r corresponds to the aperture angle of the cone formed by the origin and the black circle.

Figure A.1. A random rotation cube and the rotation of a random 3D point by all angle-axis vectors on the surface of that cube. Observe that the rotation vector that maximises the angle $\angle(\mathbf{R}_{\mathbf{r}} \mathbf{p}, \mathbf{R}_{\mathbf{r}_0} \mathbf{p})$ lies on a cube edge. Also observe that rotation vectors on the face of the cube (grey lines in the projection) do not rotate the point beyond the convex hull of the point rotated by the edges. Best viewed in colour.

Proof. Equation (5) can be derived as follows:

$$dA = \frac{-1}{\sqrt{1 - (\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}(\lambda)}^\top \mathbf{f})^2}} d((\mathbf{R}_{\mathbf{r}_0}^{-1} \mathbf{f}) \cdot (\mathbf{R}_{\mathbf{r}(\lambda)}^{-1} \mathbf{f})) \quad (6)$$

$$= \frac{-1}{\sqrt{1 - (\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}(\lambda)}^\top \mathbf{f})^2}} \mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} d(\mathbf{R}_{\mathbf{r}(\lambda)}^\top \mathbf{f}) \quad (7)$$

$$= \frac{-1}{\sqrt{1 - (\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}(\lambda)}^\top \mathbf{f})^2}} \mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} d(\mathbf{R}_{-\mathbf{r}(\lambda)} \mathbf{f}) \quad (8)$$

$$= \frac{-1}{\sqrt{1 - (\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}(\lambda)}^\top \mathbf{f})^2}} \mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \left(-\mathbf{R}_{-\mathbf{r}(\lambda)} [\mathbf{f}]_\times \frac{(-\mathbf{r}(\lambda))(-\mathbf{r}(\lambda))^\top + (\mathbf{R}_{-\mathbf{r}(\lambda)}^\top - I)[-\mathbf{r}(\lambda)]_\times}{\|-\mathbf{r}(\lambda)\|^2} \right) d(-\mathbf{r}(\lambda)) \quad (9)$$

$$= \frac{-1}{\sqrt{1 - (\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}(\lambda)}^\top \mathbf{f})^2}} \mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \left(-\mathbf{R}_{\mathbf{r}(\lambda)}^\top [\mathbf{f}]_\times \frac{\mathbf{r}(\lambda)\mathbf{r}(\lambda)^\top - (\mathbf{R}_{\mathbf{r}(\lambda)} - I)[\mathbf{r}(\lambda)]_\times}{\|\mathbf{r}(\lambda)\|^2} \right) d(-\mathbf{r}(\lambda)) \quad (10)$$

$$= \frac{-1}{\sqrt{1 - (\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}(\lambda)}^\top \mathbf{f})^2}} \mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \left(\mathbf{R}_{\mathbf{r}(\lambda)}^\top [\mathbf{f}]_\times \frac{\mathbf{r}(\lambda)\mathbf{r}(\lambda)^\top - (\mathbf{R}_{\mathbf{r}(\lambda)} - I)[\mathbf{r}(\lambda)]_\times}{\|\mathbf{r}(\lambda)\|^2} \right) (\mathbf{r}_j - \mathbf{r}_i) d\lambda \quad (11)$$

where (9) uses Result 1 from [3] and the derivative follows from the differential. \square

Since the derivative is computationally expensive to calculate, we only evaluate it at the vertices. In addition, we only require the sign of the result, which simplifies the equation. Corollary A.1 presents the relevant results.

Corollary A.1. (*Sign of the derivative of the rotation angle function at the vertices*) Given a unit 3D bearing vector \mathbf{f} and a rotation cube \mathcal{C}_r centred at \mathbf{r}_0 with vertices $\{\mathbf{r}_i\}_{i \in [1,8]}$, then

$$\operatorname{sgn} \left. \frac{dA}{d\lambda} \right|_{\lambda=0} = \operatorname{sgn} -\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}_i}^\top [\mathbf{f}]_\times \left(\mathbf{r}_i \mathbf{r}_i^\top - (\mathbf{R}_{\mathbf{r}_i} - I) [\mathbf{r}_i]_\times \right) (\mathbf{r}_j - \mathbf{r}_i) \quad (12)$$

and

$$\operatorname{sgn} \left. \frac{dA}{d\lambda} \right|_{\lambda=1} = \operatorname{sgn} -\mathbf{f}^\top \mathbf{R}_{\mathbf{r}_0} \mathbf{R}_{\mathbf{r}_j}^\top [\mathbf{f}]_\times \left(\mathbf{r}_j \mathbf{r}_j^\top - (\mathbf{R}_{\mathbf{r}_j} - I) [\mathbf{r}_j]_\times \right) (\mathbf{r}_j - \mathbf{r}_i). \quad (13)$$

B. Tighter Upper Bound: Implementation Details

The upper bound given in Theorem 3 requires the evaluation of $\Gamma(\mathbf{f}, \mathbf{p})$ for a given translation cuboid \mathcal{C}_t . If the ray through \mathbf{f} passes through the cube $\mathbf{p}_t^{\mathbf{r}_0}$ for $\mathbf{t} \in \mathcal{C}_t$, the angle $\angle(\mathbf{f}, \mathbf{p}_t^{\mathbf{r}_0})$ is zero. Otherwise, $\angle(\mathbf{f}, \mathbf{p}_t^{\mathbf{r}_0})$ is equal to the angle between the \mathbf{f} and the point on the skeleton of the cube (vertices and edges) with least angular displacement from \mathbf{f} . Thus, for the translation domain \mathcal{C}_t with skeleton $\mathcal{S}k_t$,

$$\Gamma(\mathbf{f}, \mathbf{p}) = \begin{cases} \max_{\mathbf{t} \in \mathcal{S}k_t} \mathbf{1}(\theta - \angle(\mathbf{f}, \mathbf{p}_t^{\mathbf{r}_0}) + \psi_r(\mathbf{f}, \mathcal{C}_r)) & \text{if } \angle(\mathbf{f}, \mathbf{p}_t^{\mathbf{r}_0}) > \psi_t(\mathbf{p}, \mathcal{C}_t) \\ 1 & \text{else.} \end{cases} \quad (14)$$

The key here is finding $\angle(\mathbf{f}, \mathbf{p}_t^{\mathbf{r}_0})$ which maximises Γ over the skeleton. For the first case, the implementation needs to find $\mathbf{p}_t^{\mathbf{r}_0}$ with least angular displacement from \mathbf{f} . We use the following technique:

- (i) find octant of $\mathbf{p} - \mathbf{t}_0$ and project the cube to the unit sphere as a spherical hexagon;
- (ii) determine in which lune of the spherical hexagon (induced by the spherical hexagon) $\mathbf{R}_{\mathbf{r}_0}$ resides; and
- (iii) solve for point on hexagon edge in that lune with least angular displacement from $\mathbf{R}_{\mathbf{r}_0}$.

We know from the data structure design that all translated points $\mathbf{p} - \mathbf{t}$ for $\mathbf{t} \in \mathcal{C}_t$ lie entirely in one octant of \mathbb{R}^3 . Finding the octant (i) enables us to project the cube to a spherical hexagon on the unit sphere. That is, we can determine the 6 vertices of the cube that form the vertices of the spherical hexagon after projection and the order in which they cycle. This has simplified the problem to one of finding the closest point on the spherical hexagon to the rotated bearing vector. Finding which lune the rotated bearing lies in (ii) further simplifies the problem to one of finding the closest point on a geodesic to the rotated bearing vector. This can be solved in closed form (iii).

C. Precomputing Angles on the Sphere

The nested structure of this implementation may be exploited by pre-computing the angle between the translated 3D points and any location on the unit sphere. Thus, for a fixed translation, the angle between any rotated bearing vector and its rotationally-closest 3D point may be pre-computed. This is essentially the analogue of a distance transform on the surface of the sphere S^2 , in that the sphere is discretised and a look-up table constructed. By using this precomputation, the max operation in (27) and (28) is reduced from $\mathcal{O}(M)$ to $\mathcal{O}(1)$.

The procedure for constructing this look-up table follows. We subdivide the sphere into 98304 regions by projecting it onto an enclosing cube whose faces are partitioned with quad-trees. The configuration chosen ensures that the maximum angle between an arbitrary point and its nearest cell centre is 0.6° . We use a linear projection onto the cube, which facilitates rapid conversion from a unit vector to a location in the data structure. The disadvantage of a linear projection is that the cell sizes are not uniform, although they all respect the angle condition given above.

Mathematically, the effect of using this data structure is to relax θ by up to 0.6° . This means that the result is no longer optimal with respect to θ , although it should be near-optimal. We did not use this feature in the experiments, because it cannot ensure optimality, although it is useful for solving problems with large M , that is, many 3D points.

D. Comparison with Weaker Bounds

The weaker sphere-based uncertainty angles ψ_r^w and ψ_t^w given in (7) and (16) appeared originally in [4] and [1] respectively. The tighter cuboid-based uncertainty angles ψ_r and ψ_t given in (10) and (13) are original to this work and lead to tighter bounds on the objective function. This can be seen from Theorems 1 and 2, where it is clear that $\bar{f} - \underline{f}$ is smaller when the uncertainty angles ψ_r and ψ_t are smaller. The proofs that $\psi_r \leq \psi_r^w$ and $\psi_t \leq \psi_t^w$ will now be given.

Lemma D.1. (*Rotation uncertainty angle inequality*) Given a 3D bearing vector \mathbf{f} and a rotation cube \mathcal{C}_r of half-side length δ_r centred at \mathbf{r}_0 , then

$$\psi_r(\mathbf{f}, \mathcal{C}_r) \leq \psi_r^w(\mathcal{C}_r). \quad (15)$$

Proof. Inequality (15) can be derived as follows:

$$\psi_r(\mathbf{f}, \mathcal{C}_r) = \min \left\{ \max_{\mathbf{r} \in S_r} \angle(\mathbf{R}_r \mathbf{f}, \mathbf{R}_{\mathbf{r}_0} \mathbf{f}), \pi \right\} \quad (16)$$

$$= \min \left\{ \angle(\mathbf{R}_{\mathbf{r}^*} \mathbf{f}, \mathbf{R}_{\mathbf{r}_0} \mathbf{f}), \pi \right\} \quad (17)$$

$$\leq \min \left\{ \sqrt{3}\delta_r, \pi \right\} \quad (18)$$

$$= \psi_r^w(\mathcal{C}_r) \quad (19)$$

where (17) replaces the maximisation with the arg max rotation \mathbf{r}^* and (18) follows from Lemma 2. \square

Lemma D.2. (*Translation uncertainty angle inequality*) Given a 3D point \mathbf{p} and a translation cuboid \mathcal{C}_t of half-side length δ_t centred at \mathbf{t}_0 , then

$$\psi_t(\mathbf{p}, \mathcal{C}_t) \leq \psi_t^w(\mathbf{p}, \mathcal{C}_t). \quad (20)$$

Proof. Inequality (20) can be derived as follows. For $\|\mathbf{p} - \mathbf{t}_0\| \geq \rho_t$, which is guaranteed for $\rho_t \leq \zeta$,

$$\psi_t(\mathbf{p}, \mathcal{C}_t) = \max_{\mathbf{t} \in \mathcal{V}_t} \angle(\mathbf{p} - \mathbf{t}, \mathbf{p} - \mathbf{t}_0) \quad (21)$$

$$\leq \max_{\mathbf{t} \in S_t^2} \angle(\mathbf{p} - \mathbf{t}, \mathbf{p} - \mathbf{t}_0) \quad (22)$$

$$= \arcsin \left(\frac{\rho_t}{\|\mathbf{p} - \mathbf{t}_0\|} \right) \quad (23)$$

$$= \psi_t^w(\mathbf{p}, \mathcal{C}_t) \quad (24)$$

where (22) follows from maximising the angle over the circumsphere S_t^2 of the cuboid instead of the vertices and (23) is shown in [1] with ρ_t being the half space diagonal of the translation subcuboid \mathcal{C}_t . For $\|\mathbf{p} - \mathbf{t}_0\| < \rho_t$, $\psi_t \leq \pi = \psi_t^w$. \square

Therefore, the weaker uncertainty angles ψ_r^w and ψ_t^w are larger than the uncertainty angles ψ_r and ψ_t for a given rotation or translation subcuboid. Consequently, the bounds using ψ_r and ψ_t are tighter than those using ψ_r^w and ψ_t^w .

E. Proof of Algorithm Convergence

A requirement of branch-and-bound is that the upper and lower bounds converge as the size of the branch tends to zero. The convergence of the bounds can be proved as follows. It is clear that the upper bound (19) is equal to the lower bound (17) when the uncertainty angles $\psi_r(\mathbf{f}, \mathcal{C}_r)$ and $\psi_t(\mathbf{p}, \mathcal{C}_t)$ are zero. Similarly, the tighter upper bound (22) is equal to the lower bound when the rotation uncertainty angle $\psi_r(\mathbf{f}, \mathcal{C}_r)$ is zero and the translation subcuboid \mathcal{C}_t is of size zero, since then $\angle(\mathbf{f}, \mathbf{p}_{\mathbf{t}_0}^{\mathbf{r}_0}) = \angle(\mathbf{f}, \mathbf{p}_{\mathbf{t}_0}^{\mathbf{r}_0})$ for $\mathbf{t} \in \mathcal{C}_t$. It remains to be seen that $\psi_r(\mathbf{f}, \mathcal{C}_r)$ and $\psi_t(\mathbf{p}, \mathcal{C}_t)$ tend to zero as the size of the subcuboids \mathcal{C}_r and \mathcal{C}_t tend to zero, irrespective of the value of \mathbf{f} or \mathbf{p} .

The rotation uncertainty angle $\psi_r(\mathbf{f}, \mathcal{C}_r)$ involves a maximisation over all rotations on the surface of the subcube \mathcal{C}_r . As the subcube size tends to zero, in the limit the surface and centre of the cube become identified and therefore the angle $\angle(\mathbf{R}_r \mathbf{f}, \mathbf{R}_{\mathbf{r}_0} \mathbf{f})$ equals zero. The translation uncertainty angle $\psi_t(\mathbf{p}, \mathcal{C}_t)$ involves a maximisation over all translations on the vertices of the subcuboid \mathcal{C}_t . As the subcuboid size tends to zero, in the limit the vertices and centre of the cuboid become identified and therefore the angle $\angle(\mathbf{p} - \mathbf{t}, \mathbf{p} - \mathbf{t}_0)$ equals zero. The point \mathbf{p} cannot lie inside the subcuboid for a sufficiently small cuboid, since the translation domain has been restricted to exclude translations for which $\|\mathbf{p} - \mathbf{t}\| < \zeta$. Therefore the upper and lower bounds converge as the size of subcuboids (branches) tend to zero.

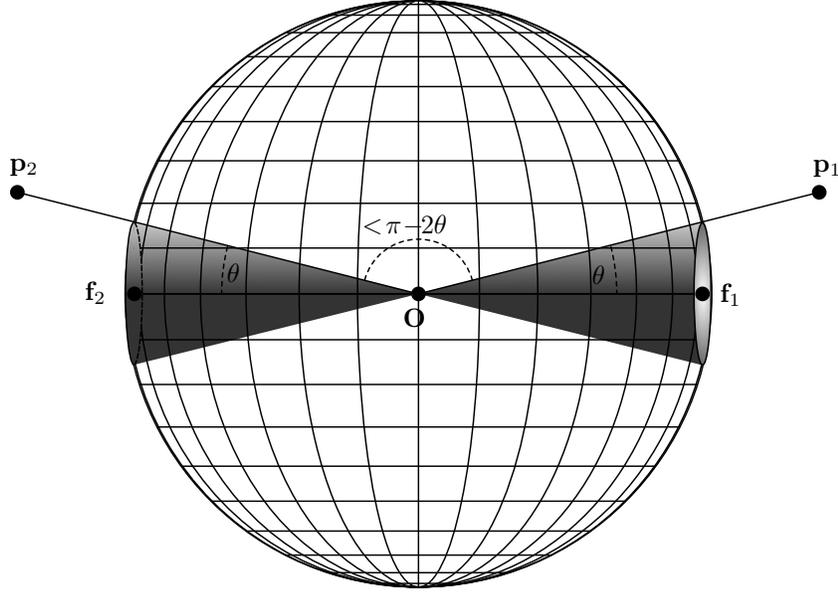


Figure E.1. Example of a critical configuration (rotation-only). The angle between the 3D point vectors $\angle(\mathbf{p}_1, \mathbf{p}_2)$ is infinitesimally less than $\pi - 2\theta$. To prove that the maximum number of inliers is 1, infinitesimally small rotation subcubes will be required.

However, an advantage of the inlier maximisation formulation is that the gap between the bounds becomes exactly zero well before the branch size becomes infinitesimal. There are nonetheless critical configurations of points and bearing vectors for which the bounds will only converge in the limit. The simplest case is illustrated in Figure E.1. In this rotation-only example, the angle between the 3D point vectors is infinitesimally less than $\pi - 2\theta$. To prove that the maximum number of inliers is 1, infinitesimally small rotation subcubes will be required.

In order to guarantee that the algorithm terminates in finite time, a small tolerance value η must be subtracted from the uncertainty angles. That is, replace the uncertainty angles in all the formulae with their primed versions: $\psi'_r = \psi_r - \eta$ and $\psi'_t = \psi_t - \eta$. For the tighter upper bound, η also has to be added to $\angle(\mathbf{f}, \mathbf{p}_t^{\mathbf{r}_0})$. Writing the nested bounds with the uncertainty angle tolerances gives the following. The upper bound $\bar{\nu} \triangleq \bar{\nu}_t$ (19) for the translation cuboid \mathcal{C}_t is found by running rotation BB until convergence with the following bounds

$$\underline{\nu}_r \triangleq \sum_{\mathbf{f} \in \mathcal{F}} \max_{\mathbf{p} \in \mathcal{P}} \mathbf{1}(\theta - \angle(\mathbf{f}, \mathbf{p}_{\mathbf{t}_0}^{\mathbf{r}_0}) + \psi'_t(\mathbf{p})) \quad (25)$$

$$\bar{\nu}_r \triangleq \sum_{\mathbf{f} \in \mathcal{F}} \max_{\mathbf{p} \in \mathcal{P}} \mathbf{1}(\theta - \angle(\mathbf{f}, \mathbf{p}_{\mathbf{t}_0}^{\mathbf{r}_0}) + \psi'_t(\mathbf{p}) + \psi'_r(\mathbf{f})). \quad (26)$$

The tighter upper bound (22) instead uses

$$\underline{\nu}_r \triangleq \sum_{\mathbf{f} \in \mathcal{F}} \max_{\mathbf{p} \in \mathcal{P}, \mathbf{t} \in \mathcal{C}_t} \mathbf{1}(\theta - \angle(\mathbf{f}, \mathbf{p}_{\mathbf{t}}^{\mathbf{r}_0}) - \eta) \quad (27)$$

$$\bar{\nu}_r \triangleq \sum_{\mathbf{f} \in \mathcal{F}} \max_{\mathbf{p} \in \mathcal{P}, \mathbf{t} \in \mathcal{C}_t} \mathbf{1}(\theta - \angle(\mathbf{f}, \mathbf{p}_{\mathbf{t}}^{\mathbf{r}_0}) - \eta + \psi'_r(\mathbf{f})). \quad (28)$$

The lower bound $\underline{\nu} \triangleq \underline{\nu}_t$ (17) for \mathcal{C}_t is found by running rotation BB until convergence using bounds (25) and (26) with ψ'_t set to zero. That is, the bounds

$$\underline{\nu}_r \triangleq \sum_{\mathbf{f} \in \mathcal{F}} \max_{\mathbf{p} \in \mathcal{P}} \mathbf{1}(\theta - \angle(\mathbf{f}, \mathbf{p}_{\mathbf{t}_0}^{\mathbf{r}_0})) \quad (29)$$

$$\bar{\nu}_r \triangleq \sum_{\mathbf{f} \in \mathcal{F}} \max_{\mathbf{p} \in \mathcal{P}} \mathbf{1}(\theta - \angle(\mathbf{f}, \mathbf{p}_{\mathbf{t}_0}^{\mathbf{r}_0}) + \psi'_r(\mathbf{f})). \quad (30)$$

In this work, η was set to machine epsilon for maximal precision. In C++, this can be accessed by using the command `std::numeric_limits<float>::epsilon()`.

F. Time Complexity

Explicitly including the tolerance η in the bound formulae makes it possible to derive a bound on the worst-case search tree depth and thereby obtain the time complexity of the algorithm. In terms of the size of the input, the GOPAC algorithm is $O(MN)$, or $O(N)$ if angle precomputation is used, where M is the number of 3D points and N is the number of bearing vectors. However, the notation conceals a very large constant. Including the constant factors that can be selected by the user yields $O(\rho_{t_0}^3 \zeta^{-3} \eta^{-6} MN)$, where ρ_{t_0} is the half space diagonal of the initial translation cuboids, that is one-quarter the space diagonal of the translation domain, and ζ and η are small previously-defined constants set by the user.

Calculating the upper and lower bounds involves a summation over \mathcal{F} and a maximisation over \mathcal{P} , therefore the complexity is $O(MN)$. If angle precomputation is used, the maximisation becomes a constant-time lookup leading to a bound complexity of $O(N)$. However, it is as of yet unclear how the number of iterations (explored subcuboids) depends on the inputs. The central finding is that branch-and-bound is exponential in the worst-case tree search depth D , but D is logarithmic in η^{-1} . Therefore the complexity of BB is polynomial in η^{-1} , where η is the angle tolerance. Rotation and translation search will be treated separately before being combined into an analysis of nested rotation and translation search.

Theorem 1. (Rotation Search Depth and Complexity) Let $\rho_{r_0} = \sqrt{3}\delta_{r_0} = \sqrt{3}\pi/2$ be the half space diagonal of the initial rotation subcube \mathcal{C}_{r_0} . Then

$$D_r = \max \left\{ \left\lceil \log_2 \frac{\rho_{r_0}}{\eta} \right\rceil, 0 \right\} \quad (31)$$

is an upper bound on the worst-case rotation tree search depth for an uncertainty angle tolerance η and $O(\eta^{-3})$ is the time complexity of rotation BB search.

Proof. Rotation BB converges when $\underline{\nu}_r \geq \bar{\nu}_r$. For any $\psi'_t(\mathbf{p}, \mathcal{C}_t)$ in (25) and (26), $\underline{\nu}_r \geq \bar{\nu}_r$ when $\psi'_r(\mathbf{f}, \mathcal{C}_r) \leq 0$ or equivalently $\psi_r(\mathbf{f}, \mathcal{C}_r) \leq \eta$ for all $\mathbf{f} \in \mathcal{F}$. Now,

$$\psi_r(\mathbf{f}, \mathcal{C}_r) = \min \left\{ \max_{\mathbf{r} \in \mathcal{S}_r} \angle(\mathbf{R}_r \mathbf{f}, \mathbf{R}_{\mathbf{r}_0} \mathbf{f}), \pi \right\} \quad (32)$$

$$= \min \left\{ \angle(\mathbf{R}_{\mathbf{r}^*} \mathbf{f}, \mathbf{R}_{\mathbf{r}_0} \mathbf{f}), \pi \right\} \quad (33)$$

$$\leq \min \left\{ \sqrt{3}\delta_r, \pi \right\} \quad (34)$$

$$\leq \rho_r \quad (35)$$

where (33) replaces the maximisation with the $\arg \max$ rotation \mathbf{r}^* , (34) follows from Lemma 2 and ρ_r is the half space diagonal of the rotation subcube \mathcal{C}_r . At rotation search tree depth D_r , the half space diagonal is given by

$$\rho_{r_{D_r}} = \frac{1}{2} \rho_{r_{D_r-1}} = \frac{1}{2^{D_r}} \rho_{r_0}. \quad (36)$$

Substituting into (35) gives

$$\psi_r(\mathbf{f}, \mathcal{C}_{r_{D_r}}) \leq \rho_{r_{D_r}} = 2^{-D_r} \rho_{r_0}. \quad (37)$$

To find the worst-case rotation search tree depth, the constraint $\psi_r(\mathbf{f}, \mathcal{C}_r) \leq \eta$ is applied:

$$\psi_r(\mathbf{f}, \mathcal{C}_{r_{D_r}}) \leq 2^{-D_r} \rho_{r_0} \leq \eta. \quad (38)$$

Taking the logarithm of both sides yields

$$D_r \geq \log_2 \frac{\rho_{r_0}}{\eta}. \quad (39)$$

Equation (31) follows from the requirement that D_r be a non-negative integer. Now, rotation BB will have examined at most

$$N_r = 8(1 + 8 + 8^2 + \dots + 8^{D_r}) = 8 \frac{8^{D_r+1} - 1}{8 - 1} = \frac{8}{7} ((2^{D_r+1})^3 - 1) \quad (40)$$

subcubes at search depth D_r , due to the octree structure. Finally, substituting (31) into (40) and simplifying using Bachmann–Landau notation gives

$$N_r = O\left(\left(\frac{\rho_{r_0}}{\eta}\right)^3\right) = O(\eta^{-3}). \quad (41)$$

The ρ_{r_0} term is removed because it is a constant (equal to $\sqrt{3}\pi/2$) that is not selected by the user. \square

The analysis of the worst-case search depth and time complexity for translation search proceeds in a similar manner.

Theorem 2. (*Translation Search Depth and Complexity*) Let ρ_{t_0} be the half space diagonal of the initial translation subcuboid \mathcal{C}_{t_0} . Then

$$D_t = \max\left\{\left\lceil \log_2 \frac{\rho_{t_0}}{\zeta \sin \eta} \right\rceil, 0\right\} \quad (42)$$

is an upper bound on the worst-case translation tree search depth for an uncertainty angle tolerance η and $O(\rho_{t_0}^3 \zeta^{-3} \eta^{-3})$ is the time complexity of translation BB search.

Proof. Translation BB converges when $\nu_t \geq \bar{\nu}_t$. This condition is met when $\psi'_t(\mathbf{p}, \mathcal{C}_t) \leq 0$ or equivalently $\psi_t(\mathbf{p}, \mathcal{C}_t) \leq \eta$ for all $\mathbf{p} \in \mathcal{P}$. This can be seen by inspecting (25) and (29) and noting that at convergence the upper and lower rotation bounds will be equal. Now for $\|\mathbf{p} - \mathbf{t}_0\| \geq \rho_t$, which is guaranteed for $\rho_t \leq \zeta$,

$$\psi_t(\mathbf{p}, \mathcal{C}_t) = \max_{\mathbf{t} \in \mathcal{V}_t} \angle(\mathbf{p} - \mathbf{t}, \mathbf{p} - \mathbf{t}_0) \quad (43)$$

$$\leq \max_{\mathbf{t} \in S_t^2} \angle(\mathbf{p} - \mathbf{t}, \mathbf{p} - \mathbf{t}_0) \quad (44)$$

$$= \arcsin\left(\frac{\rho_t}{\|\mathbf{p} - \mathbf{t}_0\|}\right) \quad (45)$$

$$\leq \arcsin\left(\frac{\rho_t}{\zeta}\right) \quad (46)$$

where (44) follows from maximising the angle over the circumsphere S_t^2 of the cuboid instead of the vertices, (45) is shown in [1] with ρ_t being the half space diagonal of the translation subcuboid \mathcal{C}_t , and (46) follows from the restriction of the translation domain such that $\|\mathbf{p} - \mathbf{t}\| \geq \zeta$. At translation search tree depth D_t , the half space diagonal of $\mathcal{C}_{t_{D_t}}$ is given by

$$\rho_{t_{D_t}} = \frac{1}{2} \rho_{t_{D_t-1}} = \frac{1}{2^{D_t}} \rho_{t_0}. \quad (47)$$

Substituting into (46) gives

$$\psi_t(\mathbf{p}, \mathcal{C}_{t_{D_t}}) \leq \arcsin\left(\frac{\rho_{t_{D_t}}}{\zeta}\right) = \arcsin\left(\frac{\rho_{t_0}}{\zeta 2^{D_t}}\right). \quad (48)$$

To find the worst-case translation search tree depth, the constraint $\psi_t(\mathbf{p}, \mathcal{C}_t) \leq \eta$ is applied, resulting in

$$\psi_t(\mathbf{p}, \mathcal{C}_{t_{D_t}}) \leq \arcsin\left(\frac{\rho_{t_0}}{\zeta 2^{D_t}}\right) \leq \eta. \quad (49)$$

Taking the sine and logarithm of both sides yields

$$D_t \geq \log_2 \frac{\rho_{t_0}}{\zeta \sin \eta}. \quad (50)$$

Equation (42) follows from the requirement that D_t be a non-negative integer. Now, translation BB will have examined at most

$$N_t = 8(1 + 8 + 8^2 + \dots + 8^{D_t}) = 8 \frac{8^{D_t+1} - 1}{8 - 1} = \frac{8}{7} ((2^{D_t+1})^3 - 1) \quad (51)$$

subcuboids at search depth D_t . Finally, substituting (42) into (51) and simplifying using Bachmann–Landau notation and the Taylor expansion of $\sin \eta$ gives

$$N_t = O(\rho_{t_0}^3 \zeta^{-3} (\sin \eta)^{-3}) = O(\rho_{t_0}^3 \zeta^{-3} \eta^{-3}). \quad (52)$$

\square

In the nested BB search structure detailed at the beginning of Section 5, for every translation subcuboid examined, rotation BB search is run once to find the lower translation bound and again to find the upper translation bound. Thus the number of rotation subcubes examined is at worst equal to $2N_t N_r$. For each rotation subcube, both the upper and lower bounds must be calculated, each with a time complexity of $O(MN)$. Thus the total number of bound calculations is at worst equal to $4N_t N_r$. Combining the time complexity analyses (41) and (52) with the time complexity of the bound calculations leads to the following corollary.

Corollary F.1. (*Time Complexity of GOPAC*) Let ρ_{t_0} be the half space diagonal of the initial translation subcuboid C_{t_0} , ζ be the translation restriction parameter, η be the uncertainty angle tolerance, M be the number of 3D points and N be the number of bearing vectors, then the time complexity of the GOPAC algorithm is given by

$$O(\rho_{t_0}^3 \zeta^{-3} \eta^{-6} MN). \quad (53)$$

It is important to observe that experimental evaluation of runtime is more revealing for BB algorithms than time complexity analysis. The main reason to use BB is that it can prune large regions of the search space, reducing the size of the problem. This is not reflected in the complexity analysis.

G. Additional Experiments and Results

G.1. Synthetic Data Experiments: Cube Prior

In the paper, we applied a torus prior in order to be as fair to the local optimisation approaches as possible. However, in general, restrictive constraints on the rotation and translation cannot be easily obtained. A more realistic prior, that can still be used by BlindPnP [7] and SoftPOSIT [2] in order to facilitate comparison, is a cube prior. This constrains the camera centre to a cube with no restriction on rotation. The intent of this prior is to simulate the task of camera pose estimation with respect to a scene, such as locating a camera inside a building, instead of the torus prior which simulates the task of camera pose estimation with respect to an object, such as a teapot on a table.

We use a cube centred at a random location in $[-1, 1]^3$ with side-length 0.5m. Again, we use the Monte Carlo simulation framework of [7] for the experiments: M random 3D points were generated from $[-1, 1]^3$; a fraction ω_{3D} of the 3D points were randomly selected as outliers to model occlusion; the inliers were projected to a virtual image 640×480 with an effective focal length of 800 and centre of (320, 240) and normal noise was added with $\sigma = 2$ pixels; and random points were added to the image such that a fraction ω_{2D} of the 2D points were outliers.

BlindPnP represents this pose prior with a 50 component Gaussian mixture model, the means of which are used to initialise SoftPOSIT, as in [7]. We have increased the number of components from the 20 recommended by the authors to help model the increased rotation uncertainty. The mixture model was trained from a set of 200 random camera centres in the cube and a set of 200 uniform random rotation matrices, generated using the method in [6].

The results are shown in Figure G.1. Two success rates are reported: the fraction of trials where the true maximum number of inliers was found and the fraction where the correct pose was found, where the angle between the output rotation and the ground truth rotation is less than 0.1 radians and the camera centre error $\|\mathbf{t} - \mathbf{t}_{GT}\| / \|\mathbf{t}_{GT}\|$ relative to the ground truth \mathbf{t}_{GT} is less than 0.1, as in [7]. The 2D and 3D outlier fractions were fixed to 0 when not being varied and multithreading was used in the 2D outlier experiments. Our method (GOPAC) outperforms the other methods by a significant margin. In particular, SoftPOSIT was unable to find the correct pose in all experiments, largely due to being unable to handle 3D points behind the camera. BlindPnP is also sensitive to 3D outliers behind the camera and is relatively slow with 50 mixture model components, as is reflected in its results. However, it was necessary to use 50 in order to obtain reasonable results for the camera pose. Moreover, without a strong rotation prior, both local methods have difficulty finding the best pose.

G.2. Synthetic Data Experiments: Early Termination

The early termination strategy, referred to in the paper as “truncated GOPAC” or [GP], is investigated further in this section. Recall that since the majority of the runtime of the algorithm is spent decreasing the upper bound, an early termination strategy will often attain the global optimum, although it will not be able to guarantee optimality. We repeated the 2D outlier experiments for the torus prior with random points to show the performance of our algorithm when it is terminated after 30s. At termination, the algorithm returns the best-so-far cardinality and camera pose, as well as a flag to indicate that the result is not guaranteed to be optimal. The results are shown in Figure G.2. It is clear that the method still performs very well even when terminated early, albeit without an optimality guarantee. For some applications, it may be worth sacrificing optimality for the significant decrease in runtime. For comparison, we also plot the results from the paper without truncation.

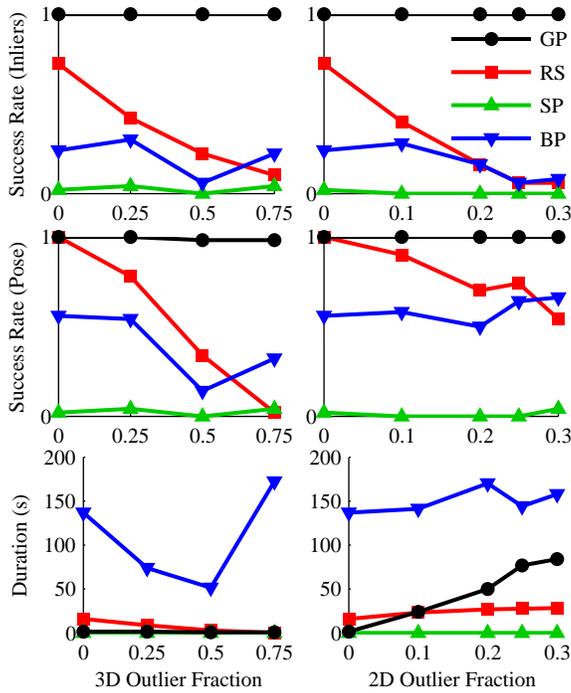


Figure G.1. Cube Prior. Mean success rates and median runtimes with respect to the 3D and 2D outlier fractions for the random points dataset, for 50 Monte Carlo simulations per parameter value with the cube prior. The 2D outlier fraction was fixed at 0 for the 3D outlier experiments and the 3D outlier fraction was fixed at 0 for the 2D outlier experiments.

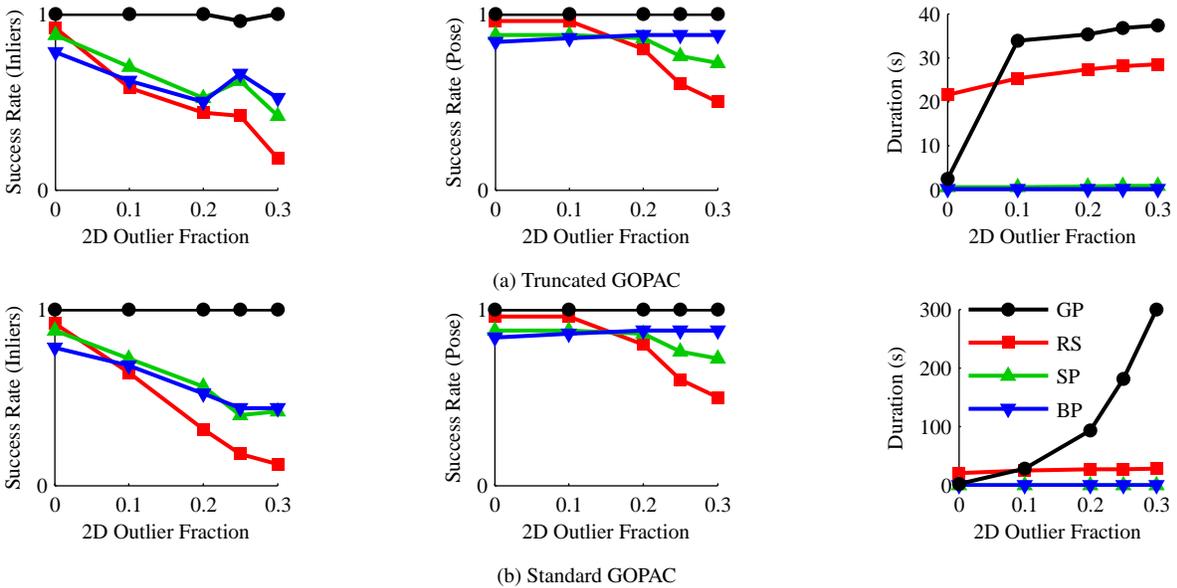


Figure G.2. A comparison of “truncated GOPAC” and standard GOPAC. Mean success rates and median runtimes with respect to the 2D outlier fractions for the random points dataset, for 50 Monte Carlo simulations per parameter value with the torus prior. The 3D outlier fraction was fixed at 0.

G.3. Real Data Experiments

In this section, we present quantitative and qualitative results for a new dataset and additional quantitative and qualitative results for the experiments reported in Section 6.2. The new dataset was generated from a different scene (“Scene5”) of the outdoor DATA61/2D3D dataset, using the same method described in Section 6.2. It consists of a 3D point-set with 98 points, a set of 11 images containing 30 2D features and a set of ground truth camera poses. In contrast to the residential scene reported in the main paper (“Scene1”), this scene is mixed-use, with residential and commercial or industrial areas, is less

Table G.1. Camera pose results for the supplementary DATA61/2D3D dataset (“Scene5”). Quartiles for translation error, rotation error and runtime and the mean inlier recall and success rates are reported. [GP] denotes truncated GOPAC, where search is terminated after 30s, with no optimality guarantee. RS_K denotes RANSAC with K million iterations.

Method	GP	[GP]	RS_{20}	RS_{240}
Translation Error (m)	1.59 / 2.03 / 11.2	1.70 / 2.72 / 11.3	7.79 / 31.5 / 45.8	1.74 / 4.11 / 14.9
Rotation Error ($^\circ$)	2.14 / 3.28 / 179	1.95 / 3.70 / 179	117 / 179 / 179	3.52 / 140 / 179
Recall (Inliers)	1.00	0.98	0.78	0.86
Success Rate (Inliers)	1.00	0.55	0.00	0.09
Success Rate (Pose)	0.55	0.55	0.18	0.45
Runtime (s)	229 / 346 / 409	33 / 33 / 33	29 / 29 / 32	339 / 347 / 382

Table G.2. Camera pose results for the DATA61/2D3D dataset (“Scene1”). Quartiles for translation error, rotation error and runtime and the mean inlier recall and success rates are reported. [GP] denotes truncated GOPAC, where search is terminated after 30s, with no optimality guarantee. RS_K denotes RANSAC with K million iterations.

Method	GP	[GP]	RS_{20}	RS_{280}
Translation Error (m)	1.77 / 2.30 / 4.37	1.85 / 3.10 / 6.31	13.0 / 20.3 / 24.8	19.2 / 28.5 / 38.4
Rotation Error ($^\circ$)	1.75 / 2.08 / 3.15	1.92 / 3.04 / 137	90.2 / 178 / 179	117 / 179 / 179
Recall (Inliers)	1.00	0.97	0.75	0.81
Success Rate (Inliers)	1.00	0.45	0.00	0.00
Success Rate (Pose)	0.82	0.64	0.09	0.09
Runtime (s)	311 / 477 / 496	33 / 34 / 34	33 / 34 / 35	453 / 471 / 480

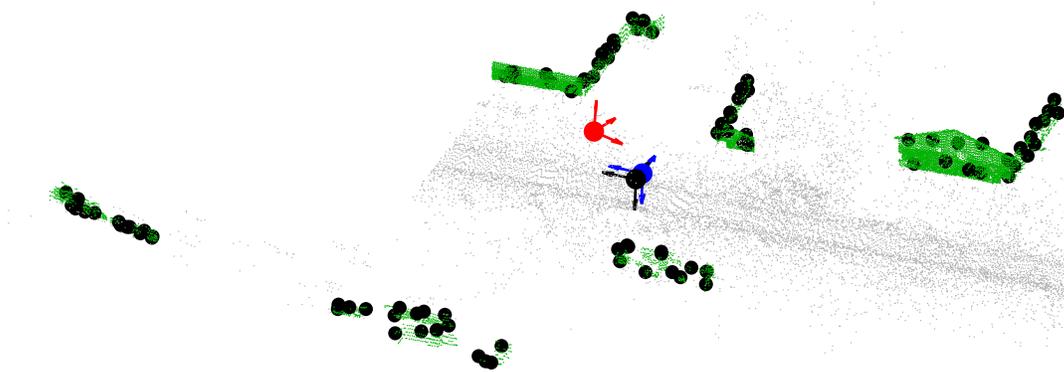
spread out, and contains fewer buildings. For this experiment, we used an inlier threshold of $\theta = 2^\circ$, multithreading and a 2D outlier fraction guess of $\omega_{2D} = 0.25$. The translation domain was $50 \times 5 \times 5$ m, covering two lanes of the road, making use of the knowledge that the camera was mounted on a survey vehicle.

Qualitative results for the GOPAC and RANSAC algorithms are shown in Figure G.3 and quantitative results in Table G.1. GOPAC finds the optimal number of inliers for all frames and the correct camera pose for the majority of frames, despite the naïvety of the 2D/3D point extraction process. It is clear however that finding the optimal inlier set does not always correspond to finding the optimal pose for such a weak feature extraction procedure. In contrast, RANSAC performed better on this dataset than the original one, since it contains fewer, denser 3D clusters. This is beneficial for RANSAC because it reduces the implicitly-searched pose space, whereas it is disadvantageous for GOPAC because it does not reduce the search space and can mean that the inlier set at the correct pose has a lower cardinality than the inlier set at some incorrect poses. The main failure modes for GOPAC and RANSAC were 180° rotation flips, due to ambiguities arising from the low angular separation of points in the vertical direction.

Due to space restrictions, some additional quantitative and qualitative results for the experiments reported in Section 6.2 were not presented in the main paper. In Table G.2, Table 1 has been rewritten with quartile information for a more complete representation of the data. In Figure G.4, Figure 1 has been re-plotted at a larger scale. In Figures G.5, G.6 and G.7, we plot the remaining 10 images with the results obtained by GOPAC and RANSAC. It can be seen that there are two failure cases, with respect to camera pose (both remain optimal with respect to the number of inliers). In both cases, some of the extracted 2D features lie on non-building pixels, due to an error in the segmentation, which can be thought of as particularly undesirable 2D outliers. This is likely to have contributed to the algorithm finding the incorrect pose.

References

- [1] M. Brown, D. Windridge, and J.-Y. Guillemaut. Globally optimal 2D-3D registration from points or lines without correspondences. In *Proc. 2015 Int. Conf. Comput. Vision*, pages 2111–2119, 2015. 4, 7
- [2] P. David, D. Dementhon, R. Duraiswami, and H. Samet. SoftPOSIT: simultaneous pose and correspondence determination. *Int. J. Comput. Vision*, 59(3):259–284, 2004. 8
- [3] G. Gallego and A. Yezzi. A compact formula for the derivative of a 3-D rotation in exponential coordinates. *J. Mathematical Imaging Vision*, 51(3):378–384, 2015. 3
- [4] R. I. Hartley and F. Kahl. Global optimization through rotation space search. *Int. J. Comput. Vision*, 82(1):64–79, 2009. 4
- [5] J. Kiefer. Sequential minimax search for a maximum. *Proc. Am. Math. Soc.*, 4(3):502–506, 1953. 1
- [6] F. Mezzadri. How to generate random matrices from the classical compact groups. *Notices Amer. Math. Soc.*, 54(5):592–604, 2007. 8
- [7] F. Moreno-Noguer, V. Lepetit, and P. Fua. Pose priors for simultaneously solving alignment and correspondence. In *Proc. 2008 European Conf. Comput. Vision*, pages 405–418. Springer, 2008. 8

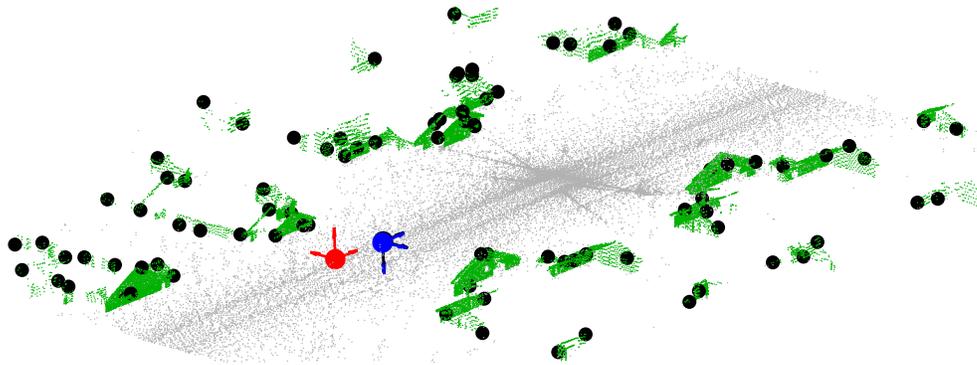


(a) 3D point-set (grey and green), 3D features (black dots) and ground-truth (black), RANSAC (red) and our (blue) camera poses. The ground-truth and our camera poses coincide, whereas the RANSAC pose has a translation offset and a 180° rotation offset. Best viewed in colour.



(b) Panoramic photograph and extracted 2D features (top), building points projected onto the image using the RANSAC camera pose (middle) and building points projected using our camera pose (bottom).

Figure G.3. Qualitative camera pose results for the supplementary DATA61/2D3D dataset (“Scene5”). The pose of the camera when it captured image 6 is shown, as is the projection of the building points onto image 6.

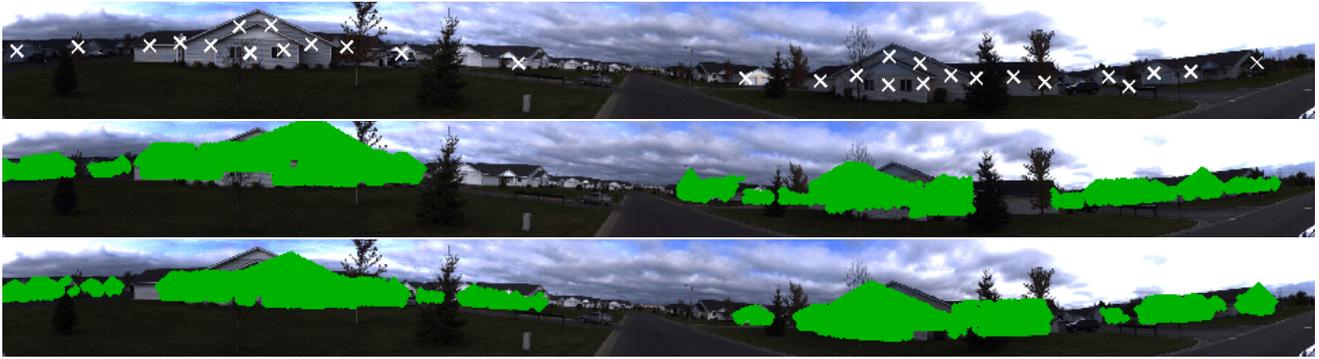


(a) 3D point-set (grey and green), 3D features (black dots) and ground-truth (black), RANSAC (red) and our (blue) camera poses. The ground-truth and our camera poses coincide, whereas the RANSAC pose has a translation offset and a 180° rotation offset. Best viewed in colour.



(b) Image 10. Panoramic photograph and extracted 2D features (top), building points projected onto the image using the RANSAC camera pose (middle) and building points projected using our camera pose (bottom).

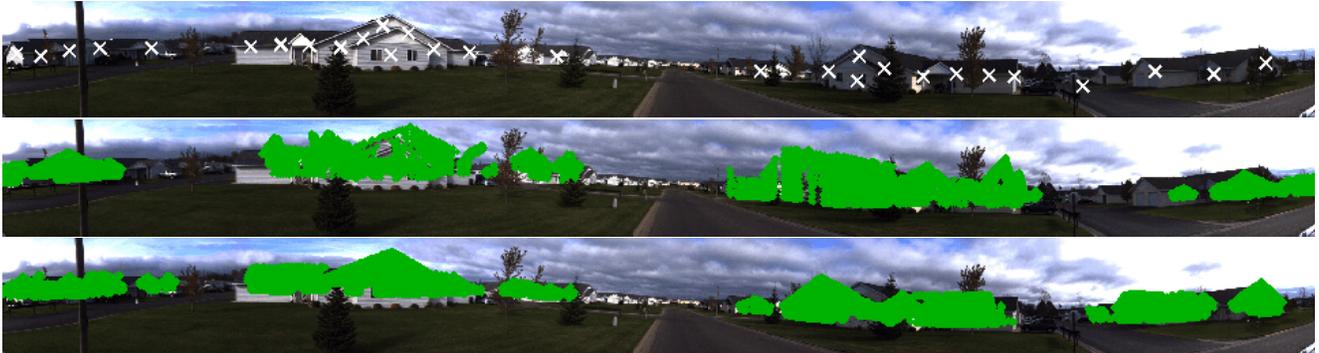
Figure G.4. Estimating the pose of a calibrated camera from a single image within a large-scale, unorganised 3D point-set captured by vehicle-mounted laser scanner. Our method solves the absolute pose problem while simultaneously finding feature correspondences, using a globally-optimal branch-and-bound approach with tight novel bounds on the cardinality of the inlier set.



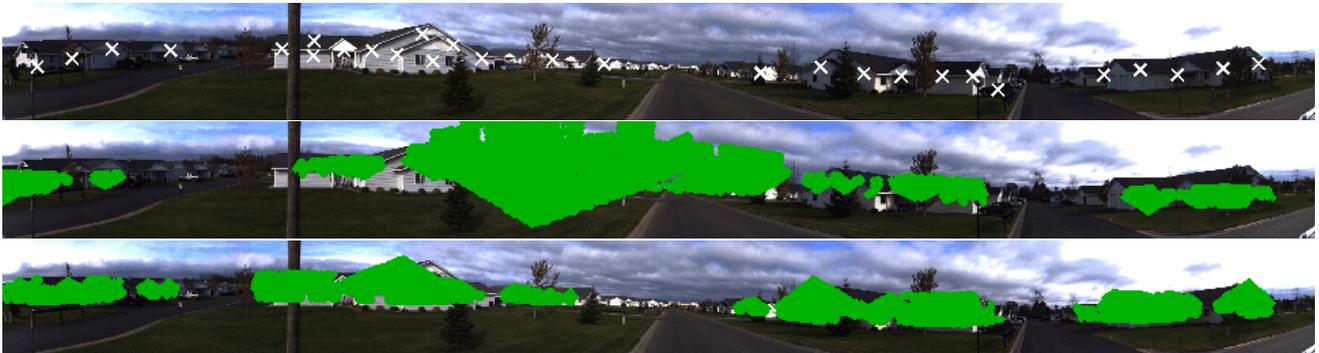
(a) Image 1: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).



(b) Image 2: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).



(c) Image 3: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).

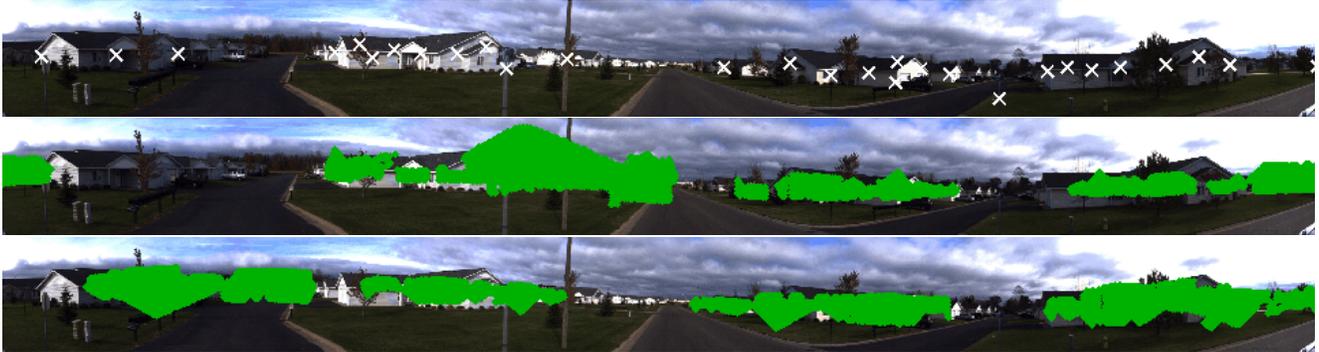


(d) Image 4: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).

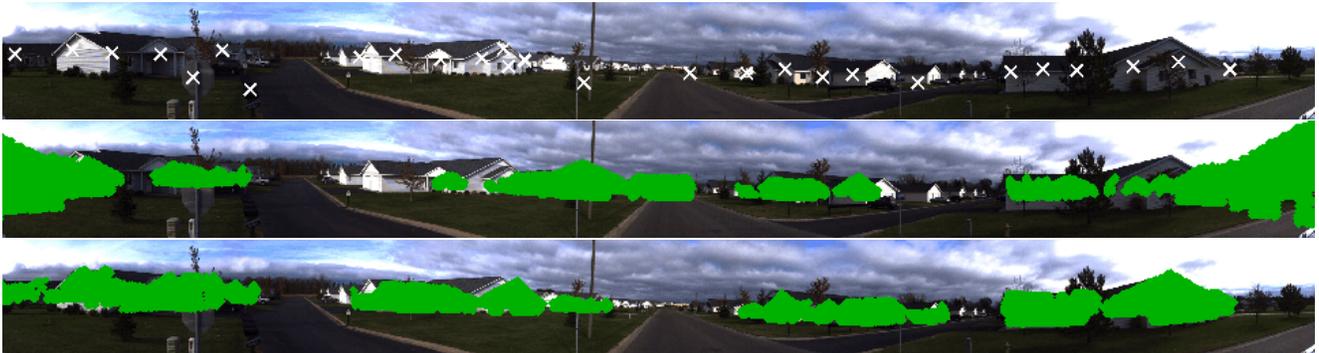
Figure G.5. Qualitative results for 4 test images. Our method (GOPAC) found the correct camera pose for every image.



(a) Image 5: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).

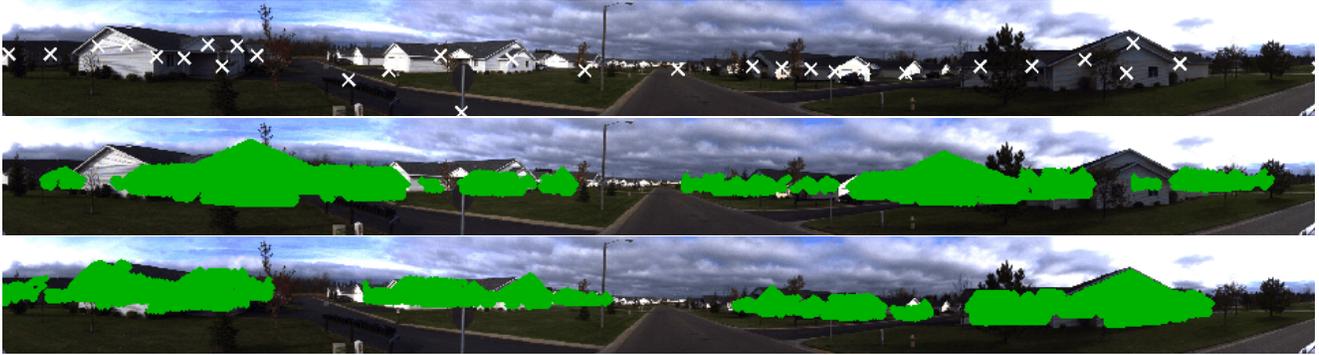


(b) Image 6: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).

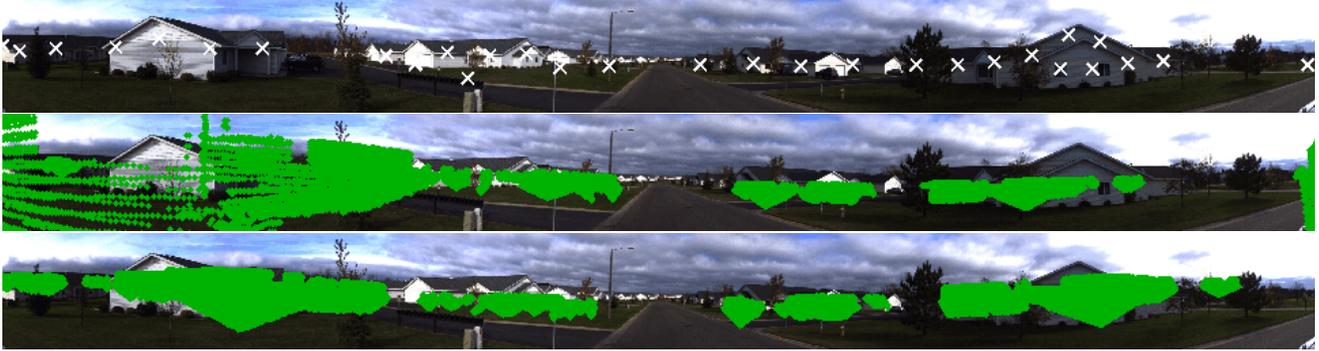


(c) Image 7: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).

Figure G.6. Qualitative results for 3 test images. Image 6 was a failure case for our method, with respect to camera pose, although it still found the optimal number of inliers. It can be seen that three 2D features were extracted at non-building locations, due to an error in the segmentation. This may have contributed to the algorithm finding an incorrect pose.



(a) Image 8: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).



(b) Image 9: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).



(c) Image 11: 2D features (top), building points projected using RANSAC camera pose (middle) and building points projected using our pose (bottom).

Figure G.7. Qualitative results for 3 test images. Image 9 was a failure case for our method, with respect to camera pose, although it still found the optimal number of inliers. Like the previous failure case, 2D features were extracted at non-building locations, due to an error in the segmentation. This may have contributed to the algorithm finding an incorrect pose.