Globally Optimal 2D-3D Registration from Points or Lines Without Correspondences

Mark Brown[1], David Windridge[1,2], and Jean-Yves Guillemaut[1]
[1]CVSSP, University of Surrey, Guildford, GU2 7XH, United Kingdom
[2]School of Science and Technology, Middlesex University, London, NW4 4BT, United Kingdom

Abstract

We present a novel approach to 2D-3D registration from points or lines without correspondences. While there exist established solutions in the case where correspondences are known, there are many situations where it is not possible to reliably extract such correspondences across modalities, thus requiring the use of a correspondence-free registration algorithm. Existing correspondence-free methods rely on local search strategies and consequently have no guarantee of finding the optimal solution. In contrast, we present the first globally optimal approach to 2D-3D registration without correspondences, achieved by a Branch-and-Bound algorithm. Furthermore, a deterministic annealing procedure is proposed to speed up the nested branch-and-bound algorithm used. The theoretical and practical advantages this brings are demonstrated on a range of synthetic and real data where it is observed that the proposed approach is significantly more robust to high proportions of outliers compared to existing approaches.

1. Introduction

In this paper we present a globally optimal solution to the *2D-3D* registration problem with points or lines when *no correspondences* between features are known, *and* in the presence of outliers. It is an integral part of the more general *2D-3D* registration problem: given an image taken by a calibrated camera and a 3D model, the objective is to determine the pose of the camera with respect to the model. This finds use in a range of applications, e.g. motion segmentation [26], and object localisation and recognition [17].

The general *2D-3D* registration problem is challenging since, while there exist techniques to extract features between 2D and 3D (e.g. corners [14], salient features [6] or lines [13, 29]), it is an open problem to automatically establish correspondences between them. This may be due to a variety of reasons. In the case of lines, there are many scenes where it is difficult to establish correspondences based on appearance, for example in highly repetitive man-made scenes or where low-width structures are present [16]. Furthermore, feature appearance can vary dramatically between 3D and its 2D projection due to the non-linear nature of the transformation; a 3D feature may be projected from a large range of viewpoints and perspective distortion may occur. More generally however, correspondences of any feature type are particularly difficult to hypothesize when the 3D model is untextured, as is often the case if it is obtained by a laser range scanner.

The lack of feature correspondences renders traditional hypothesize-and-test approaches (e.g. RANSAC [12]) practically obsolete due to the very high computational complexity of the problem. Whilst some progress has been made to compute solutions more efficiently [10, 25], existing approaches all suffer from the same limitations: they only search for local maxima and hence require a good initialisation with no guarantee of optimality given.

The paper makes the following contributions. Firstly, we propose the first globally optimal solution to this problem, achieved via a *Branch-and-Bound* (BnB) strategy. Its formulation readily allows for either point or line features to be used, allowing it to be applicable to a broader range of scenes. Secondly, we propose a novel *deterministic annealing* parameter that allows for the speed-up of nested BnB algorithms while preserving the global optimality of the solution. Finally, the approach is evaluated against state-of-the-art where significant improvements are demonstrated: our approach is more accurate and significantly more robust to high rates of outliers compared to existing approaches.

The structure of this paper is as follows: in Section 2 an overview of related work is given. In Section 3 the scope of the problem is formally defined and in Section 4 the Branch-and-Bound approach is detailed, including derivations required for this problem and our novel deterministic annealing strategy. In Section 5 results are presented for points and lines on synthetic and real data. Finally, conclusions and future work are presented in Section 6.

2. Related Work

Initially, a review is given of proposed solutions to the *2D-3D* registration problem from points or lines without correspondences. We are not aware of any that explicitly use points *or* lines within the same framework. With respect to points, one of the best, early approaches is the *SoftPosit* algorithm [10]. It iterates between determining the pose and determining multiple, weighted correspondences, allowing for a larger basin of convergence than approaches that only allow binary assignment of correspondences (e.g. ICP). More recently, Moreno-Noguer *et al.* [25] propose a solution to the same problem, known as *BlindPnP*, by modelling the initial set of poses as a Gaussian Mixture Model and using each component to initialise a Kalman filter. It performs comparably to *SoftPosit* in a similar amount of time, except in large amounts of clutter, where *SoftPosit* is outperformed by *BlindPnP*. Both of these approaches search for local maxima and hence often do not obtain the global optimum.

With respect to pose estimation from lines, an early solution is proposed in [3] who use a local search procedure to iteratively arrive at local optima. Alternatively, the *SoftPosit* algorithm has been extended to use lines [11] by minimising the distance between the endpoints of the 3D line and the infinitely long 2D line. Bhat and Heikkilä [4] systematically sample and rank the space of potential poses however it is computationally inefficient for large numbers of lines. Other approaches are more restrictive, e.g. based on graph matching [8] where the graph structure is often not preserved under a full projective transformation, or assume a number of 3D lines are orthogonal [9].

Branch-and-bound solutions to geometry estimation in computer vision haven been proposed for a number of different problems, typically requiring novel derivations of bounds in each case. The earliest approaches are due to Breuel [5] who focuses mainly on 2D-2D registration with up to 4 degrees of freedom. He derives geometrically meaningful bounds that describe the maximum distance a feature can move by under a bounded set of transformations. Hartley and Kahl [15] derived bounds for the group of 3D rotations, allowing for globally optimal relative pose estimation [15], and *3D-3D* registration [30].

Alternative branch-and-bound approaches rely on *linear programming* techniques to compute bounds, e.g. [2]. In a naive form they may only be applied to linear transformations, so to be more widely applicable nonlinear constraints are relaxed into linear convex and concave envelopes (e.g. [27, 2]). Alternatively, Jurie [18] approximates perspective pose by orthographic pose to create a linear *2D-3D* registration problem that is solved by similar techniques, however its use of the Gaussian error model results in an approach that is not robust to outliers.

Our approach is the first globally optimal approach to *2D-3D* registration using either points or lines without correspondences. There are some similarities between our approach and the globally optimal *3D-3D* registration algorithm *Go-ICP* [30]. In contrast, our problem firstly requires the derivation of uniformly continuous bounds for the *2D-3D* problem. Secondly, we propose a novel deterministic annealing parameter that allows for the speed-up of nested branch-and-bound algorithms. Thirdly we propose a more general solution, extending the framework to use points *or* lines, allowing for broader scene applicability.

3. Problem Formulation

Initially we give the problem definition for 2D and 3D *features* in general, before moving onto the specifics for points or lines.

Let there be N 2D features $\{\Lambda_i\}_{i=1}^{N}$ and M 3D features $\{\Psi_j\}_{j=1}^{M}$, and denote the *distance* between a 3D and 2D feature as $d(\Psi_j, \Lambda_i)$. Where no outliers are present, the objective is to find the rotation $\mathbf{R} \in SO(3)$ and camera centre $\mathbf{C} \in \mathbb{R}^3$ that minimise the following:

$$\sum_{i=1}^{N} \min_{j \in \{1 \dots M\}} d(\mathbf{R}(\Psi_j - \mathbf{C}), \Lambda_i) \tag{1}$$

To make (1) robust to outliers, we use *trimming*: instead of minimising the sum over all 2D features it is minimised over the smallest k values, where k represents the expected number of inliers. Without loss of generality, assume the terms of the sum in (1) have been re-ordered in ascending order, yielding the *trimmed objective*: finding $\mathbf{R} \in SO(3)$ and $\mathbf{C} \in \mathbb{R}^3$ that minimise:

$$\sum_{i=1}^{k}{}^{*} \min_{j \in \{1 \dots M\}} d(\mathbf{R}(\Psi_j - \mathbf{C}), \Lambda_i) \tag{2}$$

where * denotes the sum rearranged in ascending order (note this depends upon $\mathbf{R}$ and $\mathbf{C}$).

To apply (2) for points (denoted $\Lambda_i^{(P)}$ and $\Psi_j^{(P)}$) or lines (denoted $\Lambda_i^{(L)}$ and $\Psi_j^{(L)}$) simply requires the distance measure to be defined. In the case of points, denote each 2D point by X_i and each 3D point by Y_j. For convenience, assume the 2D point has been reprojected onto the unit sphere i.e. $X_i \in \mathbb{R}^3, ||X_i|| = 1$. Then the distance between them is the *geometrically meaningful* angular distance, defined as:

$$d(\Psi_j^{(P)}, \Lambda_i^{(P)}) = \angle(Y_j, X_i) = \arccos\left(\frac{Y_j \cdot X_i}{||Y_j||}\right) \tag{3}$$

In the case of lines, a suitable distance measure is less obvious. Approaches to pose estimation from line correspondences (e.g. [1]) often decouple the problem into the determination of the rotation by using the direction of the 3D line, then determine the camera centre by using an arbitrary point on a line. Inspired by this approach, our line

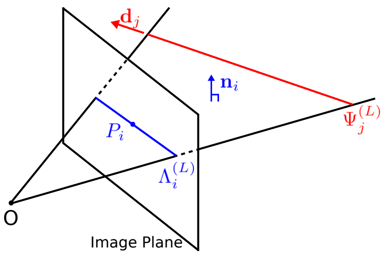

Figure 1: An illustration of the terminology used in defining a distance measure for lines. $\Lambda_i^{(L)}$ denotes a 2D line, P_i its midpoint and $\mathbf{n}_i$ the normal to its backprojected plane. $\Psi_j^{(L)}$ denotes a 3D line and $\mathbf{d}_j$ its normalised direction vector.

distance measure is as follows: for each 3D line, denote its normalised direction vector as $\mathbf{d}_j$. For each 2D line, denote its midpoint as P_i, and backproject the line, denoting the normal to this plane as $\mathbf{n}_i$ (see Figure 1 for an illustration of these terms). In the ideal, noiseless case, $\mathbf{d}_j$ will lie on the backprojected plane and P_i will lie on the projection of line $\Psi_j^{(L)}$. Hence, a suitable distance between the lines is defined as:

$$d(\Psi_j^{(L)}, \Lambda_i^{(L)}) = \lambda \left|\left| \frac{\pi}{2} - \angle(\mathbf{d}_j, \mathbf{n}_i) \right|\right| + \angle(\Psi_j^{(L)}, P_i) \quad (4)$$

where λ defines the relative weighting between the two terms and $\angle(\Psi_j^{(L)}, P_i)$ denotes the angle between P_i and the nearest point of the projected *line segment* $\Psi_j^{(L)}$. By using this we are implicitly considering 2D lines as infinitely long but 3D lines as finitely long (similar to [19]).

4. Branch-and-Bound

Branch-and-Bound (BnB) is a very general framework for global optimisation. Assume the objective is to minimise some function f over an N-dimensional bounded space $\Omega \subset \mathbb{R}^N$. Assume further that for any subset $\omega \subseteq \Omega$ (hereafter, known as a *branch*) a *lower bound* and an *upper bound* may be determined for the minimal value of f in this branch, and that these bounds converge as the size of the branch tends to zero. For example, the upper bound could simply be the value of the function at the midpoint of the branch, and the lower bound could be the upper bound minus some expression for how much the function can deviate in an interval of that size.

These assumptions allow for the determination of a solution to f whose value is within ϵ of the globally optimal solution, for *any* user-specified $\epsilon > 0$. It relies upon recursively subdividing the space, calculating upper and lower bounds for each branch. Initially the input to the algorithm is simply the branch Ω, and, at any stage in the algorithm, there is a set of branches that are subsets of Ω, each with a lower and upper bound to the minimum value f can take in that branch. At each stage of the algorithm the following two steps are performed:

1) Determine the distance between the *lowest* lower bound and *lowest* upper bound of the bounds in the set of branches. If this distance is less than ϵ the algorithm terminates, outputting the lowest upper bound and its branch.

2) Otherwise, consider the branch that has the lowest lower bound and subdivide it further, computing upper and lower bounds for each sub-branch.

The algorithm will converge because, eventually, the *size* of the branches considered will be sufficiently small that the distance between the upper bound and lower bound of a newly divided branch will be less than ϵ. When this occurs, the outputted value is within ϵ of the globally optimal solution because the entirety of Ω has been (recursively) searched and so it is known that any better solution is no more than ϵ better than the one returned.

For the *2D-3D* registration problem, optimisation takes place over the space $SE(3) = SO(3) \text{ x } \mathbb{R}^3$. This space is unbounded, so it is assumed the camera centre is known to lie within a bounded set Ω_C - typically a reasonable assumption when Ω_C encapsulates a suitably large space.

This section is structured as follows: in 4.1, we give *geometrically meaningful* bounds that describe how much the features can be transformed by within a given neighbourhood and in 4.2 how these are used to bound the objective function. Then we describe the nested BnB structure in 4.3 and our novel *deterministic annealing* strategy in 4.4. Finally, local refinement techniques are detailed in 4.5.

4.1. Geometric Bounds

Bounds are considered separately for the rotation component and camera centre component. Firstly, the rotation bound is computed. Rotations are considered in the *axis-angle representation*: a rotation is represented by a vector $\mathbf{r} \in \mathbb{R}^3$ whose direction specifies the axis of rotation and whose magnitude specifies the angle. Hence, only rotations within the sphere of radius π need to be considered. The rotation matrix that $\mathbf{r}$ represents may be computed via the equation $\mathbf{R} = \mathbf{I} + \sin(||\mathbf{r}||)[\hat{\mathbf{r}}]_\times + (1 - \cos(||\mathbf{r}||)[\hat{\mathbf{r}}]_\times^2$, where $\hat{\mathbf{r}} = \mathbf{r}/||\mathbf{r}||$ and $[\hat{\mathbf{r}}]_\times$ denotes the skew-symmetric matrix representation of $\hat{\mathbf{r}}$. The following result is due to [15]:

Lemma 1: Let $\mathbf{R}_0$, $\mathbf{R}$ be rotation matrices and $\mathbf{r}_0$, $\mathbf{r}$ their corresponding axis-angle representations. Then, for any point $\mathbf{X} \in \mathbb{R}^3$:

$$||\mathbf{r}_0 - \mathbf{r}|| < \delta_R \Rightarrow \angle(\mathbf{R}_0\mathbf{X}, \mathbf{R}\mathbf{X}) < \delta_R := \epsilon_R \quad (5)$$

In the context of BnB, if one considers a branch as a cube of rotations in their axis-angle representation where the centre of the branch is $\mathbf{r}_0$ and the cube has side-length δ_R, then by the above result, for any rotation $(\mathbf{R})$ within the cube and for any point $\mathbf{X}$, $\angle(\mathbf{R}_0\mathbf{X}, \mathbf{R}\mathbf{X}) < \frac{\sqrt{3}\epsilon_R}{2}$.

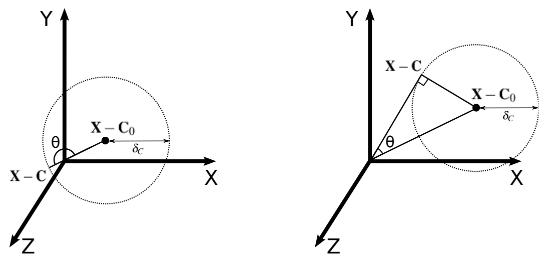

Figure 2: Left: If $\delta_C \geq ||\mathbf{X} - \mathbf{C}_0||$, the maximum angle is π by placing $\mathbf{X} - \mathbf{C}$ behind the origin. Right: If not, the maximum angle is when $\mathbf{X} - \mathbf{C}$ is at a right angle to $\mathbf{C} - \mathbf{C}_0$.

Next, bounds on the camera centre are derived.

Lemma 2: Let $\mathbf{C}_0$, $\mathbf{C} \in \mathbb{R}^3$. For any point $\mathbf{X} \in \mathbb{R}^3$, let $\theta = \angle(\mathbf{X} - \mathbf{C}_0, \mathbf{X} - \mathbf{C})$. Then:

$$\begin{aligned} &||\mathbf{C}_0 - \mathbf{C}|| < \delta_C \Rightarrow \\ &\quad \theta \leq \begin{cases} \pi & \delta_C \geq ||\mathbf{X} - \mathbf{C}_0|| \\ asin\left(\frac{\delta_C}{||\mathbf{X} - \mathbf{C}_0||}\right) & otherwise \end{cases} \\ &\quad := \epsilon_C^{\mathbf{X} - \mathbf{C}_0} \end{aligned} \tag{6}$$

Lemma 2 can be intuitively understood by referring to Figure 2. More formally:

Proof:
If $\delta_C \geq ||\mathbf{X} - \mathbf{C}_0||$, then one may construct the maximum angle of π by placing $\mathbf{X} - \mathbf{C}$ behind (or on) the origin. Otherwise, assume $\delta_C < ||\mathbf{X} - \mathbf{C}_0||$, and by the cosine rule one obtains

$$||\mathbf{X} - \mathbf{C}||^2 < 2||\mathbf{X} - \mathbf{C}_0||||\mathbf{X} - \mathbf{C}|| \cos(\theta) \tag{7}$$

hence $\cos(\theta) \geq 0$, i.e. $\theta \in [0, \frac{\pi}{2}]$. Since $\sin(\theta)$ is a strictly increasing function in this interval, obtaining an upper bound on $\sin(\theta)$ will yield an upper bound on θ. By the sine rule:

$$\sin(\theta) = \frac{||\mathbf{C}_0 - \mathbf{C}||}{||\mathbf{X} - \mathbf{C}_0||} \sin(\angle(\mathbf{C}_0 - \mathbf{C}, \mathbf{X} - \mathbf{C})) \tag{8}$$

Without loss of generality $\mathbf{X}$ and $\mathbf{C}_0$ may be assumed to be constant, hence the expression is maximised when $\angle(\mathbf{C} - \mathbf{C}_0, \mathbf{X} - \mathbf{C}) = \frac{\pi}{2}$. The result follows. ■

4.1.1 A Uniformly Continuous Bound

The function governing the bounds on the camera centre (6) is not uniformly continuous: the relationship between ϵ_C and δ_C is dependent on $\mathbf{X}$. This causes real difficulties for the algorithm: if precision ϵ_C is desired and $\mathbf{X}$ is arbitrarily close to $\mathbf{C}_0$, an arbitrarily small branch (δ_C) is required. Hence the algorithm will not converge in finite time.

To alleviate this we modify the objective function slightly so as to be uniformly continuous: when computing (2) we *only* take into account 3D features whose distance from the camera centre is larger than a specified threshold (γ). For a suitably small threshold this is sensible in practice: in general very few features will be located immediately in front of the camera.

In doing so, Equation (6) may be rewritten with γ substituted in place of $||\mathbf{X} - \mathbf{C}_0||$. This now creates a uniformly continuous function since the relationship between δ_C and ϵ_C is independent of $\mathbf{X}$. More explicitly, if a precision of $\epsilon_C \in (0, \pi)$ is desired, one may set $\delta_C = \gamma \sin \epsilon_C$ to guarantee a minimum branch size, hence guaranteeing the convergence of the algorithm. Note however that when the bound is explicitly calculated there is no need to substitute γ in place of $||\mathbf{X} - \mathbf{C}_0||$ as this simply guarantees how large a computed bound may be; smaller bounds will be obtained without this substitution.

By combining Lemmas 1 and 2, the following result is obtained:

Theorem 1: Let $\mathbf{R}_0$, $\mathbf{R}$ be rotation matrices and $\mathbf{r}_0$, $\mathbf{r}$ their corresponding axis-angle representations. Further, let $\mathbf{C}_0$, $\mathbf{C} \in \mathbb{R}^3$. Then, for any point $\mathbf{X} \in \mathbb{R}^3$:

$$\begin{aligned} &||\mathbf{r}_0 - \mathbf{r}|| < \delta_R \wedge ||\mathbf{C}_0 - \mathbf{C}|| < \delta_C \Rightarrow \\ &\qquad \angle(\mathbf{R}_0(\mathbf{X} - \mathbf{C}_0), \mathbf{R}(\mathbf{X} - \mathbf{C})) < \epsilon_R + \epsilon_C \end{aligned} \tag{9}$$

The proof follows by combining Lemmas 1 and 2 with the triangle inequality.

4.2. Function Bounds

In this subsection, the bounds in 4.1 are related to the objective functions described in Section 3. Assume we are minimising the trimmed objective (2) with the angular distance measure for point features (3). It is required to determine *upper* and *lower* bounds for (2) when the pose space $SE(3)$ is bounded. At each stage in the BnB algorithm, the pose space will be divided up into cubes, where we consider jointly a *rotation cube* centred at $\mathbf{r}_0$ of side-length δ_R and a *camera centre cube* centred at $\mathbf{C}_0$ of side-length δ_C.

To compute the *upper bound* for (2) using points (3) the objective function is simply evaluated at $(\mathbf{R}_0, \mathbf{C}_0)$. To compute the *lower bound* the expression is derived by evaluating the function at $(\mathbf{R}_0, \mathbf{C}_0)$ and subtracting the maximum amount by which the function may deviate within that branch. Denote $z(\epsilon) = \frac{\sqrt{3}}{2}(\epsilon_R + \epsilon_C)$ and hence, the lower bound is obtained as:

$$\sum_{i=1}^{k}{}^{*} \min_{j \in \{1 \ldots M\}} \max\{0, \angle(\mathbf{R}_0(Y_j - \mathbf{C}_0), X_i) - z(\epsilon)\} \tag{10}$$

The lower bound for lines (4) is derived in a similar way; the angles for each of the two terms in (4) are bounded in the same manner (by $\frac{\sqrt{3}}{2}(\epsilon_R+\epsilon_C)$), hence, subtracting them for each term gives a lower bound for the distance from the 2D line to the nearest 3D line.

4.3. Nested Branch-and-Bound

In a similar manner to [30], we use a nested BnB structure for efficiency: an outer BnB searches over the rotation space $SO(3)$ and, for each rotation branch, the upper and lower bounds are solved by an inner BnB algorithm for the camera centre. We briefly describe the computation of bounds in the inner BnB algorithm before moving onto our *novel formulation*: an annealing procedure that takes advantage of the nested structure to quickly filter unpromising branches of rotation space, whilst still determining the solution to the same level of accuracy.

Firstly, the case for determining the *upper bound* of a *rotation cube* is considered. To do so, the rotation is considered at the centre of the cube ($\mathbf{r}_0$) and the aim is to determine the minimum value of Equation (2) where $\mathbf{r}$ is fixed to $\mathbf{r}_0$ and $\mathbf{C}$ is allowed to vary. The upper bound used in the inner algorithm is simply the value of the function at that point, i.e. computed using (10) with $z(\epsilon) = 0$, with the lower bound computed using $z(\epsilon) = \frac{\sqrt{3}}{2}\epsilon_C$.

Secondly the *lower bound* of a rotation cube is considered. The same computation is performed as for the upper bound, but takes into account the maximum amount the objective function can deviate within the rotation branch. Hence, the upper bound used in the inner algorithm in this case is computed using (10) with $z(\epsilon) = \frac{\sqrt{3}}{2}\epsilon_R$; the lower bound with $z(\epsilon) = \frac{\sqrt{3}}{2}(\epsilon_R+\epsilon_C)$. For more details on using a nested BnB structure the reader is referred to [30].

4.4. Annealing Branch-and-Bound

The nested BnB structure presented in the previous section allows for a novel variant to speed up the outer BnB algorithm. In particular, the inner BnB need not be calculated to the same level of accuracy as the outer BnB early on - it only needs to when the outer BnB nears the desired accuracy. Our approach exploits this observation, resulting in a speed up factor of approximately 2.

Initially our algorithm computes the inner BnBs with an accuracy of ϵ_{init}. When the outer BnB converges (i.e. the difference between the upper and lower bounds is less than ϵ_{init}) a slightly higher level of accuracy (lower value of ϵ) is used: the inner BnBs for existing rotation branches are re-computed using this value and the algorithm continues with the new value of ϵ. The algorithm iterates in this fashion and terminates when the difference between the upper and lower bounds is less than a specified ϵ_{end}. The way ϵ is updated is performed in a *deterministic annealing* manner parameterised by β: set $\beta_{init} = -\ln(\epsilon_{init})$ and $\beta_{end} = -\ln(\epsilon_{end})$, then ϵ decreases as β *linearly increases* from β_{init} to β_{end}. It is parameterised by two constants: s, defining the step size used as β linearly increases, and n, denoting the number of steps taken from β_{init} to β_{end}.

It can be summarised in the high level pseudocode in Algorithm 1. The algorithm terminates with precision ϵ_{end},

Algorithm 1 Annealing Branch-and-Bound

```
 1: Inputs: Data, initial rotation and camera centre
    branches, $\epsilon_{end}$, $s$, $n$.
 2: Output: Optimal (within $\epsilon_{end}$) pose ($\mathbf{R}$, $\mathbf{C}$).
 3: $UB \Leftarrow \infty$                          ▷ Lowest upper bound
 4: $LB \Leftarrow 0$                               ▷ Lowest lower bound
 5: $\beta \Leftarrow -\ln(\epsilon_{end}) - ns$.              ▷ Initialise $\beta$.
 6: Put initial rotation branch in queue of branches.
 7: while ($UB - LB > \epsilon_{end}$) do
 8:     Recompute all branches with precision $\exp(-\beta)$.
 9:     Run BnB algorithm with precision $\exp(-\beta)$, using
    branches in current queue.
10:     Update $UB$, $LB$, and branches in current queue
    with results of BnB.
11:     $\beta \Leftarrow \beta + s$
12: end while
```

despite computing a large proportion of inner BnBs to a lower precision. Hence, it retains the same degree of global optimality but is typically faster than using the naive nested BnB structure. A comparison between the two is given throughout Section 5.

4.5. Local Refinement

Similarly to other BnB approaches (e.g. [30]) we locally optimise the solution whenever a promising part of the search space is found. In our case, we use two refinement algorithms: one with a large basin of convergence that does not assume correspondences between features are known, and a more precise refinement requiring known correspondences. The first refinement is called whenever a solution is within 50% of the current best solution *and* a local refinement has not been called in a neighbourhood of this point. The second refinement is called whenever a new best solution is found (similarly to [30]) and uses the correspondences given by the trimmed nearest neighbours.

For the first local refinement algorithm with a large basin of convergence we use *SoftPosit* in the case of both points and lines [10] [11]. For the second algorithm we use *EPnP* [24] for points and the approach by Kumar and Hanson [23] for lines. It should be noted that neither of these algorithms directly minimise the objective functions used here (Equations (3) and (4)) and if local refinement does not result in a better function value the algorithm will not update its best solution. Despite this, these refinement techniques allow the BnB algorithm to more efficiently find and discard local optima and concentrate on finding the global optimum.

5. Experiments

We compare the proposed approach against existing methods to *2D-3D* feature matching without correspondences. In the case of points, we compare against *SoftPosit* [10] (referred to as *SoftPositP*) and *BlindPnP* [25]. For lines a comparison is made against the extension of *SoftPosit* to lines [11], referred to as *SoftPositL*. Our approach is tested with and without the deterministic annealing variant (see Section 4.4), referred to as *BnBDA* and *BnB* respectively.

However, it is observed that *BlindPnP* relies upon the ability to use *pose priors* on where the possible camera pose may be - represented by a Gaussian Mixture Model of typically 20 components. In their experiments the pose is constrained such that the camera lies on a torus around the 3D scene. It is often unrealistic to assume such prior knowledge, and it was difficult to alter their approach to work with a significantly larger number of priors over a greater space of $SE(3)$. Therefore, for a fair comparison, our approach was altered to use these pose priors for synthetic data only.

With regards to the parameters used, we use the same for all the experiments with the exception of k (the expected number of inliers). It varies between 20% and 60% of the number of 2D features for synthetic data; for real data it is fixed to 30% and 40% of the number of 2D points and lines respectively. We use $\epsilon = 0.0025k$ for points and $\epsilon = 0.013k$ for lines. In Equation 4 we use $\lambda = 0.3$ and for the deterministic annealing we take $s = 0.1$ and $n = 30$.

5.1. Synthetic Data

In this section we use pose priors to give a fair comparison with *BlindPnP*. Our algorithm is modified to use pose priors in the following way: the input to our algorithm is a set of branches corresponding to each pose prior. The initial size of the branches is reduced to account for the fact that the pose is constrained to lie within the torus. Hence, each pose prior generates an initial rotation branch (centred at the prior) with each branch initiating its own camera centre branch (centred at the prior). Furthermore, we terminate our approach after 500 seconds if an optimal solution has not been found, outputting the best solution found so far. We perform Monte-Carlo simulations in a similar manner to [25]: firstly, we randomly generate a set of 3D features (points or lines) and randomly choose a camera position in $SE(3)$ from the torus. A proportion of these 3D features are deemed *inliers* and are projected onto the image. Noise is added to their position (the endpoints in the case of lines) of variance 2 pixels. A number of outlying 2D features are then randomly generated on the image such that the number of 2D and 3D points is equal.

We test across a range of feature sizes (40 - 90) for 20%, 40% and 60% inliers. In each case, 20 experiments were performed. The results are shown in Figure 3 for points and Figure 5 for lines, each showing the average time taken and

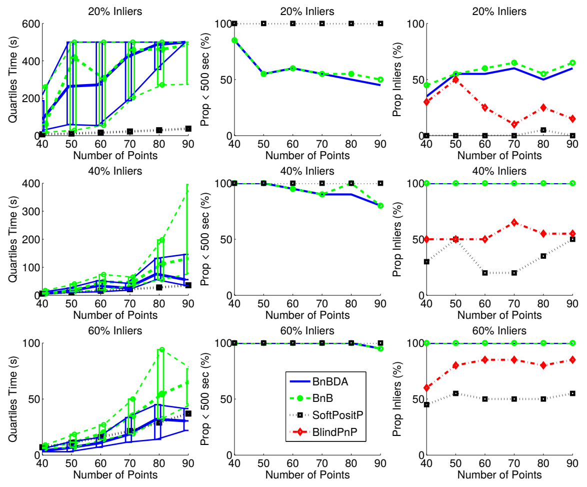

Figure 3: Time taken, proportion of solutions that converged within the time limit, and proportion of inlying solutions for each method using points (synthetic data).

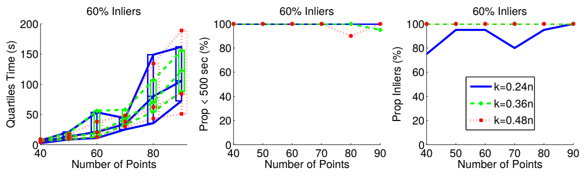

Figure 4: Results of *BnB* for increasing number of points and 60% inliers, across different values of k.

the proportion of inlying solutions. The timings for BnB algorithms have high variance, hence we report the three quartiles (on the left of Figures 3 and 5), and give the proportion of experiments that converged within the time limit (in the middle). Note that *BlindPnP* is not included in the timings - it is tested in Matlab whereas the other approaches are in C++, hence execution time comparison is not meaningful. However, Moreno-Noguer *et al.* [25] show *SoftPositP* and *BlindPnP* to have similar running times. A solution is deemed an *inlier* if the angle between its rotation ($\mathbf{R}$) and the ground truth rotation ($\mathbf{R}_{true}$) is less than a threshold (of 0.1 radians) *and* the relative error between their camera centres (expressed as $||\mathbf{C}_{true} - \mathbf{C}||/||\mathbf{C}||$) is less than a threshold (of 0.1), the same as in [25].

From these graphs it is seen that our approaches are consistently more accurate than the state-of-the-art. Interestingly, our approach sometimes does not get the right solution with 20% inliers, despite being globally optimal. It is in fact observed that, in some cases, it obtains a solution whose function value (by Equation (2)) is *lower* than the function value of the ground truth solution, despite being an outlying solution! This is indicative of the intrinsic difficulty of the problem, and the capacity of noise to redefine the global minimum. The difficulty of the problem

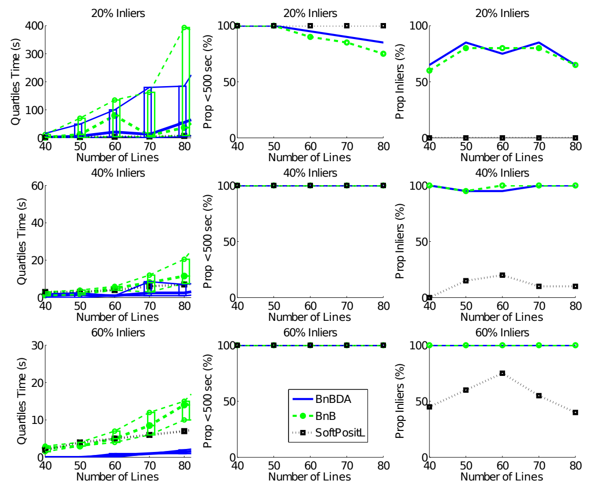

Figure 5: Time taken, proportion of solutions that converged within the time limit, and proportion of inlying solutions for each method using lines (synthetic data).

is furthermore demonstrated by the long running times, yet, the proposed globally optimal approach has a running time of only an order of magnitude higher than the heuristic approaches. It should be pointed out the code provided by [25] is used here for *BlindPnP*: it includes a termination parameter if the algorithm is taking too long. When the parameter is not used, *BlindPnP* performs more favourably for 20% inliers at the expense of significant longer execution time, however not as well as our globally optimal approach.

As a comparison between the different approaches proposed here, our annealing approach (*BnBDA*) evidently performs faster than without annealing (*BnB*), particularly so when the proportion of inliers is higher. This may be due to the function having fewer local minima when there are more inliers since it becomes easier to discard unpromising areas of space in this case. Also of interest is the fact that our algorithm is significantly faster for lines rather than points. This is a result of the objective function for lines (Equation 4): note that it is composed of two parts - the first term depends solely on the rotation while the second term depends upon both the rotation and the camera centre. It therefore allows our approach to discard unpromising areas of rotation space more quickly than in the case of points.

An important aspect to consider is the inlier ratio, which in practice cannot be known beforehand. We run experiments where we assume more conservative inlier rates than the ground truth and give results in Figure 4. When decreasing the assumed inlier rate the results do not change significantly, with only a slight decrease in accuracy when assuming 24% inliers compared to a ground truth of 60%.

We also wish to consider how long our approach takes for more input points. The following table gives this information, where it can be seen both our approach and *SoftPositP* have approximately a quadratic complexity:

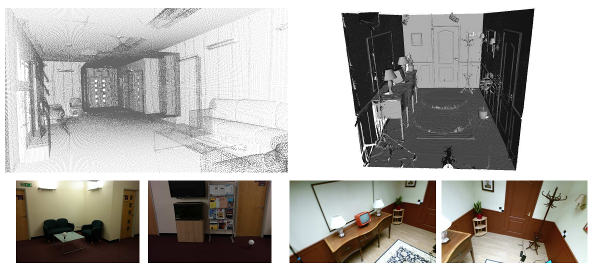

Figure 6: 3D data and example images used in our experiments. Left: *Reception*, Right: *Room*, Top: 3D data, Bottom: example images.

Table 1: Quartiles time taken (s) for 60% inliers.

Num. Pts	100	200	300	400
SoftPositP	24/24/24	102/103/103	241/242/246	403/405/408
BnBDA	39/59/85	203/210/261	749/863/1018	762/1051/1823

5.2. Real Data

5.2.1 Experimental Setup

We use real data from two datasets: *Reception* [21, 20] and *Room* [22]. Each dataset consists of both a 3D model obtained by a LiDAR scanner and a set of four and five images, see Figure 6 for an example of the data. For each image, the ground truth is estimated by manually picking point correspondences between the image and the model and using [24] to obtain the parameters. Features are obtained for *2D-3D* registration in a similar way to [25]: features are detected in a 2D image and are backprojected to the 3D model, yielding a set of 3D features. Another image that has a similar overlap of the scene is selected from which a set of 2D features is extracted. Subsequently, *2D-3D* registration is performed using the two sets of features.

Point features are detected using the *Good Features to Track* [28] algorithm - it obtains a representative small set of features for the image in a self-contained framework. Line features are detected using the LSD algorithm [13] and are reprojected to 3D using the approach by Buch *et al.* [7]. Both of these feature detection algorithms often obtain a large number of features, however the top N features may be taken using the response value provided by the algorithms. For points, 80 2D and 3D features are used, while for lines, 100 2D lines are extracted with 200 3D lines are extracted.

Data is normalised by centering the 3D data and rescaling such that all the 3D features fit inside the unit sphere. It is not necessary to do this for our approach, however it is for *SoftPosit* that uses many linear approximations. The initial branch given to our approach is the full rotation space with the camera centre branch centred at the ground truth of cube half-width 1 metre (before normalisation). Our approach is terminated after 1000 seconds if it has not already converged. *SoftPositP* is run from 4000 starting po-

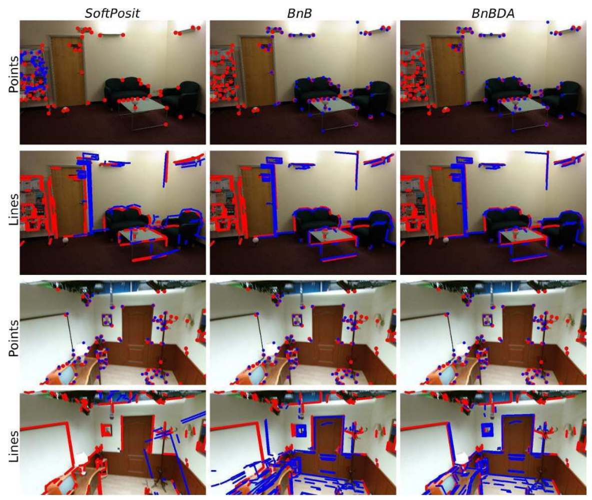

Figure 7: Example of two solutions found using *SoftPosit*, *BnB*, and *BnBDA*. The top is from *Reception* with the bottom from *Room*. 3D features are blue; 2D features are red.

sitions from the initial rotation and camera centre branch while *SoftPositL* is run from 1500, meaning all algorithms take a similar amount of time to run on our machine. Note that a comparison is not made against *BlindPnP*; their approach relies upon knowing the pose is from a small prior distribution of $SE(3)$ - this is not an assumption that can often be made in practice.

5.2.2 Results

Results are obtained for nine images. Qualitative results for four images are shown in Figure 7. Of particular note is the top-left set of results in Figure 7: evidently there are a number of outlying 2D features detected on the left of the image that were not detected in 3D. This has created a significant amount of clutter to the extent that *SoftPositP* has failed to find the optimal solution. Our globally optimal approach has recovered the correct pose in the case of both points and lines. For the other sets of results in Figure 7, our approach *always* obtains the globally optimal solution whereas *SoftPositL* fails for each case.

Now we give quantitative results. Table 2 shows, for each of the nine images, results for each method using points or lines (Note that *SP* stands for *SoftPosit*). Results are split into two parts; the top number is the angle between the estimated rotation ($\mathbf{R}$) and the ground truth rotation ($\mathbf{R}_{true}$), while the bottom number is the relative error between the camera centres ($\mathbf{C}$ and $\mathbf{C}_{true}$).

Numbers in bold represent the best result for that image. In some cases, no numbers are in bold - in this case all methods failed to return an inlying solution. In the case of both points and lines, *BnB* and *BnBDA* produced more inlying solutions than *SoftPosit* and returned more accurate results, particularly so in the case of lines.

Table 2: Quantitative Results (real data).

Image	Points			Lines		
	SP	BnB	BnBDA	SP	BnB	BnBDA
1	0.052 0.053	**0.039** **0.037**	**0.039** **0.037**	0.029 0.53	0.05 0.053	**0.03** **0.03**
2	2.6 1.2	**0.034** **0.034**	**0.034** **0.034**	0.2 0.23	0.076 0.081	**0.074** **0.078**
3	0.1 0.11	**0.083** **0.085**	**0.083** **0.085**	3.1 0.45	0.22 0.22	0.22 0.22
4	0.083 0.087	**0.066** **0.07**	**0.066** **0.07**	0.51 0.39	0.044 0.047	**0.04** **0.045**
5	1.2 1.4	3 0.93	2.5 0.46	1.9 4.7	0.35 0.43	0.24 0.35
6	0.89 0.84	2.4 0.51	2.4 0.51	2.1 1.5	0.095 **0.081**	**0.069** 0.1
7	**0.012** **0.015**	**0.012** **0.015**	**0.012** **0.015**	2.7 4.7	0.13 0.17	**0.0091** **0.01**
8	2.4 1.6	3 0.36	0.33 0.47	1.6 3.5	0.1 0.15	**0.065** **0.066**
9	**0.012** **0.016**	**0.012** **0.016**	**0.012** **0.016**	3.1 2.1	0.063 0.084	**0.044** **0.054**
Num. Inliers	4	6	6	0	5	7

The relative run-time of the methods is as follows. In the case of points, *BnB* and *BnBDA* took the maximum 1000 seconds in six out of nine images, however for the other three images *BnBDA* was approximately three times quicker than *BnB* to find the global optimum. Similar observations can be made for lines in which four out of nine images converged in less than the maximum 1000 seconds, with *BnBDA* converging on average twice as quickly as *BnB*.

6. Conclusions and Future Work

We have presented the *first* globally optimal approach to *2D-3D* registration where feature correspondences are unknown. It has been presented in a general framework rendering it applicable to either points or lines. Furthermore, a novel deterministic annealing formulation of nested BnB algorithms has been proposed, allowing for greater efficiency with no loss of optimality. It has resulted in an algorithm that is significantly better than state-of-the-art, both in terms of accuracy and robustness to high outlier rates.

Future work will include using this within a full *2D-3D* registration pipeline where 3D features are extracted directly from 3D data. This is much more challenging, requiring more repeatable features to be extracted. We furthermore plan on using both point and line features within the same framework. Points and lines are complementary feature types and it is anticipated this will result in a more robust approach that is applicable to a wider range of scenes.

Acknowledgements

This research was executed with the financial support of the EPSRC (grant number EP/K503186/1) and EU ICT FP7 project IMPART (grant number 316564).

References

[1] A. Ansar and K. Daniilidis. Linear pose estimation from points and lines. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25:578–589, 2003. 2

[2] J.-C. Bazin, H. Li, I. Kweon, C. Demonceaux, P. Vasseur, and K. Ikeuchi. A branch-and-bound approach to correspondence and grouping problems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(7):1565–1576, 2013. 2

[3] J. Beveridge and E. M. Riseman. Optimal geometric model matching under full 3d perspective. *Computer Vision and Image Understanding*, 61(3):351 – 364, 1995. 2

[4] K. K. S. Bhat and J. Heikkilä. Line matching and pose estimation for unconstrained model-to-image alignment. In *International Conference on 3D Vision (3DV)*, pages 155–162. IEEE Computer Society, 2014. 2

[5] T. M. Breuel. Implementation techniques for geometric branch-and-bound matching methods. *Computer Vision and Image Understanding*, 90:294, 2003. 2

[6] M. Brown, J.-Y. Guillemaut, and D. Windridge. A saliency-based approach to 2d-3d registration. In *Proc. International Conference on Computer Vision Theory and Applications (VISAPP)*, 2014. 1

[7] A. G. Buch, J. B. Jessen, D. Kraft, T. R. Savarimuthu, and N. Krüger. *Extended 3D Line Segments from RGB-D Data for Pose Estimation*, pages 54–65. Lecture Notes in Computer Science. 2013. 7

[8] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(8):749–764, 1995. 2

[9] P. David and D. DeMenthon. Object recognition in high clutter images using line features. In *International Conference on Computer Vision*, pages 1581–1588. IEEE Computer Society, 2005. 2

[10] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. Softposit: Simultaneous pose and correspondence determination. In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pages 698–714, 2002. 1, 2, 5, 6

[11] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. Simultaneous pose and correspondence determination using line features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 424–431, 2003. 2, 5, 6

[12] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 1

[13] R. Grompone, J. Jakubowicz, J. M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(4):722–732, April 2010. 1, 7

[14] C. Harris and M. Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988. 1

[15] R. Hartley and F. Kahl. Global Optimization through Rotation Space Search. *International Journal of Computer Vision*, 82(1):64–79, 2009. 2, 3

[16] M. Hofer, A. Wendel, and H. Bischof. Incremental line-based 3d reconstruction using geometric constraints. In *British Machine Vision Conference*. BMVA Press, 2013. 1

[17] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *International Conference on Computer Vision*, pages 102–111. IEEE Computer Society, 1987. 1

[18] F. Jurie. Solution of the simultaneous pose and correspondence problem using gaussian error model. *Computer Vision and Image Understanding*, 73(3):357–373, 1999. 2

[19] B. Kamgar-Parsi and B. Kamgar-Parsi. Matching 2d image lines to 3d models: Two improvements and a new algorithm. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2425–2432, 2011. 3

[20] H. Kim. Impart multi-modal dataset. `http://dx.doi.org/10.15126/surreydata.00807707`, 2014. 7

[21] H. Kim and A. Hilton. Influence of colour and feature geometry on multi-modal 3d point clouds data registration. In *International Conference on 3D Vision (3DV)*, pages 202–209. IEEE Computer Society, 2014. 7

[22] M. Klaudiny, M. Tejera, C. Malleson, J.-Y. Guillemaut, and A. Hilton. Scene digital cinema datasets. `http://dx.doi.org/10.15126/surreydata.00807665`, 2014. 7

[23] R. Kumar and A. Hanson. Robust methods for estimating pose and a sensitivity analysis. *CVGIP: Image Understanding*, 60(3):313 – 342, 1994. 5

[24] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal Computer Vision*, 81(2), 2009. 5, 7

[25] F. Moreno-Noguer, V. Lepetit, and P. Fua. Pose priors for simultaneously solving alignment and correspondence. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 405–418, 2008. 1, 2, 6, 7

[26] C. F. Olson. A general method for geometric feature matching and model extraction. *International Journal of Computer Vision*, 45(1):39–54, 2001. 1

[27] C. Olsson, F. Kahl, and M. Oskarsson. Branch and bound methods for euclidean registration problems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(5):783–794, 2009. 2

[28] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 7

[29] I. Stamos and P. K. Allen. 3-d model construction using range and image data. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition*. 1

[30] J. Yang, H. Li, and Y. Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *International Conference on Computer Vision*, pages 1457–1464. IEEE Computer Society, 2013. 2, 5