Caliber: Camera Localization and Calibration Using Rigidity Constraints

Albert Liu · Steve Marschner · Noah Snavely

30 October 2015

Abstract This article presents a camera calibration system, Caliber, and the underlying pose estimation problem it solves, which we call sensor localization with rigidity (SL-R).

SL-R is a constraint-satisfaction-like problem that finds a set of poses satisfying certain constraints. These constraints include not only relative pose constraints such as those found in SLAM and motion estimation problems, but also rigidity constraints: the notion of objects that are rigidly attached to each other so that their relative pose is fixed over time even if that pose is not known *a priori*. We show that SL-R is NP-hard, but give an inference-based algorithm that works well in practice.

SL-R enables Caliber, a tool to calibrate systems of cameras connected by rigid or actuated links, using image observations and information about known motions of the system. The user provides a model of the system in the form of a kinematic tree, and Caliber uses our SL-R algorithm to generate an estimate for the rigidity constraints, then performs nonlinear optimization to produce a solution that is locally least-squares optimal in terms of reprojection error. In this way, Caliber is able to calibrate a variety of setups that would have previously required special-purpose code to calibrate. We demonstrate Caliber in a number of different scenarios using both synthetic and experimental data.

Keywords calibration · complexity · localization

The final publication is available at Springer via `http://dx.doi.org/10.1007/s11263-015-0866-1`. Funding for this work was provided by National Science Foundation grant IIS-1011919, by the Intel Science and Technology Center for Visual Computing, and by a gift from Autodesk.

A. Liu
Department of Computer Science
Cornell University
Tel.: +408-695-3210
E-mail: ajul@cs.cornell.edu

1 Introduction

Many computer vision applications—including shape and appearance capture, vision for robots and autonomous vehicles, motion tracking for film production, and human-computer interaction systems—require calibration of systems of cameras.

In the simplest case, one is only interested in the intrinsic parameters of cameras such as focal length. This can be achieved by capturing a set of views of a calibration target such as a checkerboard, and solving for independent poses for each view along with camera intrinsics. However, in many other cases, the camera system has additional internal geometric constraints. The simplest example is a stereo pair: the relative pose between the two cameras remains constant, and learning the relative pose is an important goal of the calibration process. At the same time, the rigid link between the cameras provides an additional constraint that can be used to improve the accuracy of the calibration solution. Nonlinear optimization algorithms can optimize over such relative poses, but they require a good initial guess in order to succeed. Although specialized methods have been developed for various specific systems, such methods are limited to calibrating a single type of setup, requiring that a new method be developed whenever a new type arises.

Rigidity constraints and SL-R. In this article, we observe that the core geometric problem underlying this

kind of calibration can be expressed using **rigidity constraints**: the notion that two objects can be rigidly attached to each other, so that their relative pose is fixed over time even if that pose is not known *a priori*. Based on this observation, we introduce the sensor localization with rigidity problem (**SL-R**), in which the goal is to find a set of absolute poses that satisfies a set of relative pose constraints and rigidity constraints. We show that SL-R is NP-hard, but give a inference-based algorithm for SL-R that performs well in practice.

CALIBER. We then proceed to describe CALIBER, a calibration system built around our SL-R algorithm. CALIBER begins with a kinematic tree, which describes the structure of the calibration setup and the measurements taken. The kinematic tree consists of a set of nodes, representing objects, and a set of observations that describe the relative poses and rigidity constraints between the objects when measurements were taken. For cameras, these relative pose estimates as well as initial intrinsic estimates can come from *e.g.* the method of Zhang (1999).

CALIBER takes the kinematic tree and uses our SL-R algorithm to generate an estimate for the rigidity constraints. Finally, it performs nonlinear optimization over the extrinsics and intrinsics to produce a locally least-squares optimal solution to the calibration problem in terms of reprojection error. The use of rigidity constraints allows CALIBER to deliver greater generality, usability, and accuracy.

Generality. Previous methods either only support relative pose estimates and not rigidity constraints, or are specialized for a particular type of calibration setup and require the user to develop a new method for solving any different or novel case.

In contrast, CALIBER is able to handle these rigidity constraints, and only requires a description of the calibration setup in order to solve the problem. This relieves the user from the responsibility of working out a procedure for solving each new calibration problem.

Usability. Rigidity constraints also allow CALIBER to take input in a form that is easy for the user to express, and produce output in a form that the user is directly interested in. We present the user with a representation called a **kinematic tree**, which can be thought of as a scene graph, as a transformation hierarchy, or as a generalization of the kinematic chains used to describe robot arms. This representation is equivalent in power to SL-R, but provides a simpler and more intuitive input format to the user. In turn, this makes it easier for the user both to describe a calibration setup and to interpret the solution produced by CALIBER. Furthermore, the kinematic tree defines a specific, unambiguous, and minimal parameterization of the relative poses of the calibration setup, which is important for our optimization stage.

Accuracy. Finally, rigidity constraints allow CALIBER to use a minimal set of relative pose parameters, instead of having a separate absolute pose for every measurement. This avoids overfitting, making the underlying optimization problem fundamentally better conditioned, which results in more accurate calibration results.

Evaluation. We demonstrate CALIBER on a variety of multi-camera setups, from established problems like multiview stereo rigs to new problems, such as self-calibration of the relative pose of the LCD display and the camera on a mobile device. We also discuss how the calibration results can be evaluated to learn about the numerical strength of a calibration setup.

Source code for CALIBER is publicly available at `http://www.cs.cornell.edu/projects/caliber`.

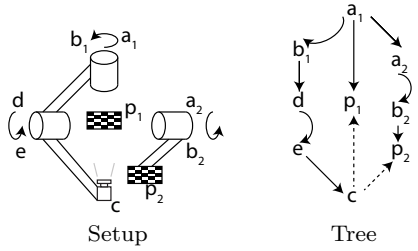


Fig. 1 Representation of a two-arm spherical gantry. The second arm has only a single axis of rotation. See Figure 4 for more examples.

1.1 Case study: spherical gantry

A particular example, which was the original motivation for the creation of CALIBER, is calibrating goniometric rigs used for appearance measurements, such as spatially-varying BRDF or BTF measurements. These rigs accomplish these measurements by automatically moving a camera and a light source to controlled positions. An archetypical example is the Stanford Spherical Gantry, the progenitor of various similar instruments. This gantry has a camera arm with two rotational degrees of freedom and a light source arm with one rotational degree of freedom, with the object to be measured placed on a central stage. A diagram of

a spherical gantry can be found in Figure 1. The goal of calibrating such a gantry is to produce a model that determines the viewing ray and light source position given any pixel position and rotation angles of the various axes. CALIBER can calibrate such a model using only views from the camera of targets located in the frame of the central stage and targets located in the light source's frame, and it easily adapts to a variety of setups that involve multiple cameras, multiple lenses, multiple light sources, etc. However, CALIBER can also solve many other calibration problems, as illustrated throughout this paper, such as calibrating camera clusters with non-overlapping views.

2 Related work

2.1 Camera calibration

The most widely used camera calibration methods in machine vision are descended from Tsai's work on practical calibration from views of 2D or 3D sets of known points (1987). Later Zhang (1999) combined these ideas with self-calibration (Maybank and Faugeras, 1992) to produce a method that fits n camera poses and one set of camera intrinsics to n images of a planar target. This approach has been widely adopted, in part because of the convenience of planar targets and in part due to Bouguet's freely available implementation (2010). The problem solved by Zhang's method corresponds in CALIBER to a tree with two nodes, a camera and a calibration target. While this method produces accurate camera intrinsics, the pose estimates can be noisy, since each can only draw upon a single view.

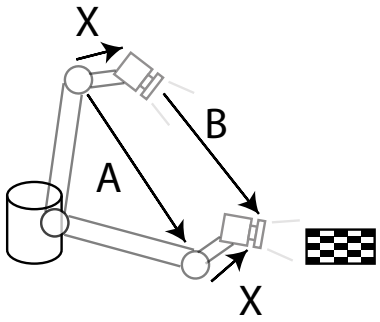


Fig. 2 The $AX = XB$ problem induced by the hand-eye calibration problem. In this case, A is the movement of the hand between two exposures (measured using the arm's encoders), B is the movement of the camera between the same two exposures (measured using the camera's views of the calibration target), and X is the unknown we wish to find, namely the relative pose between the hand and camera.

2.2 Sensor localization

The problem of determining a set of absolute poses that best fits a set of relative pose estimates has been extensively studied in various contexts. These can be represented as graphs where each vertex is an absolute pose, and edges represent relative pose estimates. In simultaneous localization and mapping (SLAM) (*e.g.* Thrun and Montemerlo, 2005), vertices correspond to positions of a robot over time and landmarks observed by the robot, while edges correspond to odometry measurements and relative pose estimates generated by the robot's views of those landmarks. Since the composition of rigid transformations around a cycle should equal the identity, cycles in the graph can lead to a more accurate solution (Estrada et al, 2009).

Similarly, structure-from-motion methods (*e.g.* Zach et al, 2010) use graphs and loop-closing to determine their alignments, and Govindu (2001; 2004) proposes an efficient method of combining the up to $\frac{n(n-1)}{2}$ possible relative pose estimates from n views.

However, while these methods are general in terms of problems involving relative pose estimates, they do not handle rigidity constraints. Kümmerle et al (2011) simultaneously produces a SLAM solution and calibrates the sensors on the robot including their relative positions, similar to the optimization step in CALIBER. However, they take as given a good initial guess for the calibration parameters and do not face the initialization problem represented by SL-R.

2.3 Rigidity constraints and $AX = XB$

Beyond localization problems involving only relative pose estimates, there are existing special-purpose methods that use solutions to a system of rigid transformation equations of the form $AX = XB$ to localize and calibrate camera systems. The oldest and best-known application of this is robot hand-eye calibration, in which the pose (relative to the hand) of a camera carried by a robot arm is established using views of a fixed calibration target. In this case, A is the movement of the hand between two exposures (measured using the arm's encoders), B is the movement of the camera between the same two exposures (measured using the camera's views of the calibration target), and X is the unknown we wish to find, namely the relative pose between the hand and camera. The robot hand-eye calibration scenario is depicted in Figure 2. This was introduced simultaneously by Shiu and Ahmad (1989) and Tsai and Lenz (1989). This problem has been an active research area for some time; see Strobl and Hirzinger (2006) for a more complete survey. The same $AX = XB$ solvers

have been used more recently (Esquivel et al, 2007) to find the relative poses of groups of rigidly connected cameras that look in different directions, such as cameras mounted on an autonomous vehicle. Even though there is no actuator in the problem, it reduces to the same $AX = XB$ problem: here A is the movement of one camera between two exposures, and B is the movement of another camera between the same two exposures.

The key common feature in these calibration problems, as well as others such the multiply-jointed robot arm of our spherical gantry setup shown in Figure 4 (e), or the facing smartphones setup (f), is rigidity. Two objects, such as cameras or robot joints, can be rigidly attached to each other. The result is that the relative pose between them remains the same as the objects move, regardless of whether the value of that relative pose is initially known. By using rigidity constraints, our algorithm subsumes these previous methods, and generalizes them so that novel systems can be calibrated, including more complex systems such as our spherical gantry that may involve solving more than one such system of equations.

Mathematically, rigidity constraints allow the same rigid transformation variable to appear multiple times in systems of $AX = XB$ equations. Our algorithm leverages existing $AX = XB$ solutions as part of its inference process. In the unambiguous case we use Park and Martin's (1994) closed-form solution which produces a least-squares optimal solution. However, it is possible for a system of $AX = XB$ equations to have an ambiguous solution; that is, the solution space for such a system may have one or more dimensions. Chen (1991) gives an analysis of the possible cases based on screw theory, and we use a set of methods based on this in the case of an ambiguous system of $AX = XB$ equations. These methods are discussed in Appendix A. Our algorithm automatically determines, selects, and solves these systems of equations based on a set of heuristics, thus freeing the user from having to work out a procedure for solving a particular instance.

2.4 Complexity

We show that SL-R, as well as its uniqueness variant, are NP-hard. Similar results were proved for structure-from-motion with missing data (Nister et al, 2007) and robot localization (Dieudonné et al, 2010), the latter reducing to the same (unique) partition problem as our proof. However, both the problem and structure of our proof are distinct from these.

3 Sensor localization with rigidity (SL-R)

Caliber ultimately computes its calibration result using a nonlinear optimization to jointly determine the internal parameters of all cameras and all unknown pose relationships in the system to be calibrated. For this nonlinear optimization to succeed, it is critical to provide a good initial starting point: an estimated solution that is approximately consistent with all the available data. Internal parameters can be estimated, one camera at a time, using existing techniques, but in many cases estimating poses requires reasoning globally about the geometry of the system. This section describes one of the key innovations of this paper: a method of finding consistent pose estimates in systems that involve rigidity constraints as well as relative pose constraints.

Abstracted away from any particular calibration setup or type of sensor, the problem boils down to approximately solving a system of constraints on a set of poses. Each pose represents the position of an element of the system, such as a camera or calibration target, at one point in time relative to some other part of the system. The constraints come in two types: **relative pose constraints** come from measurements relating two elements of the system at a particular time, and **rigidity constraints** express the fact that certain relationships remain fixed for all time. (See Section 4.4 for details of how a calibration problem is reduced to a set of constraints.)

In this section we introduce this problem, which we call sensor localization with rigidity (SL-R), and propose an algorithm that solves many instances of it. We also show in Appendix B that the general problem is NP-hard.

3.1 SL-R definition

SL-R is a constraint-satisfaction-like problem: given a set of relative pose constraints (each of which fixes the relative pose between pairs of absolute poses) and a set of rigidity constraints (each of which forces a set of relative poses to be the same), determine a set of absolute poses that fit these constraints. SL-R can be defined symbolically as follows: Subject to contraints,

$$\begin{aligned} &\text{for all} \\ &\qquad i \in \{1 \dots n\} \quad \text{(absolute pose indices)} \\ &\text{find} \\ &\qquad P_i \in SE(3) \qquad \text{(absolute poses)} \end{aligned} \tag{1}$$

fixing $P_0 = I$ as the global frame.

Constraints come in two types. First, we have relative pose constraints that force a relative pose to take

a given value T. Each such constraint has the form

given

$$(i,j) \in \{1 \dots n\}^2 \qquad \text{(relative pose index pair)}$$
$$T \in SE(3)$$

require

$$P_{ij} = P_i^{-1} P_j = T \tag{2}$$

There may be any number of such constraints.

Second, we have rigidity constraints that force all relative poses of a set L to take the same value as each other. Each such constraint has the form

given

$$L \subset \{1 \dots n\}^2$$

require

$$\forall (i,j) \in L, \forall (k,\ell) \in L \qquad P_{ij} = P_{k\ell} \tag{3}$$

Again, there may be any number of such constraints.

With only relative pose constraints, this problem is similar to that faced by SLAM and other localization problems. But the addition of rigidity constraints introduces considerably more complexity, in fact making the problem NP-hard even in the exact case as we show in Appendix B.

These equations may be underdetermined; depending on the set of constraints it may not be possible to determine a unique solution (see Section 3.2). Indeed, the underdetermined space is rich enough that the problem of determining whether a unique solution exists is *also* NP-hard. Intuitively, we can build a gadget that represeents a relative pose subproblem involving a two-fold ambiguity. This forces a choice between two options; by using rigidity constraints we can enforce that choice across the entire problem. From here it is a matter of geometrically encoding a NP-complete problem on such choices.

In practice, these equations will virtually always be overdetermined. Ideally, any instance of SL-R arising from a real scenario would have at least one solution: the actual real-world poses of the objects. However, since real-world measurements are imperfect, it is not generally possible to determine a set of absolute poses that perfectly fits a given set of measurements. We must produce a reasonable approximate solution even in the presence of noise.

We now proceed to explain the components of SL-R in more detail.

Poses. The variables of SL-R are an indexed set of n absolute poses P_i (fixing P_0 as the "root" pose representing the global coordinate frame).

Each absolute pose is a rigid transformation representing an object (such as a camera or robot joint) *at one point in time*, as given in Equation 1. By "one point in time" we are referring to the poses of the objects in the system at the moment a measurement is taken. This means that a single object could correspond to multiple different poses at different points in time; that is, when different measurements are taken. Section 4.4 will explain how these multiple poses are generated from a node in the kinematic tree.

A useful alternative expression of the poses is to consider the n^2 relative poses $P_{ij} = P_i^{-1} P_j$ between every ordered pair i, j of absolute poses, as is done in *e.g.* (Thrun and Montemerlo, 2005). These relative poses represent the pose of one object relative to another at one point in time. From this definition, the composition of relative poses around any cycle must equal the identity. This expression has the advantage of allowing us to consider each cycle to be an equation.

Relative pose constraints. Sensors such as cameras or actuator encoders can measure the relative pose between two objects. For example, cameras can measure the pose between themselves and a calibration target, whereas the value of an actuator encoder implies a particular relative pose induced by the joint. To represent these, we have relative pose constraints, which give the relative pose between two absolute poses, as given in Equation 2.

Rigidity constraints. In addition to these relative pose constraints, we have rigidity constraints. A rigidity constraint is defined by a set of relative poses. It constrains these relative poses to be the same, even if that rigid transformation is not known *a priori*, as given in Equation 3. We consider these relative poses to share a **label** and associate a common **label transformation** with that label.

Rigidity constraints represent the notion that two objects can be rigidly attached to each other. Though their absolute poses may change from measurement to measurement, the relative pose between them remains constant. Examples of this include the cameras of a rigid camera cluster, two joints at opposite ends of a rigid link, and the integrated camera of a smartphone and its screen.

Solution definition. In the exact case, the solution to an instance of SL-R assigns a rigid transformation to each absolute pose P_i (or equivalently to each relative pose P_{ij}, with the constraint that the composition of relative poses around any cycle must equal the identity) such that all of the constraints are satisfied. In

Algorithm 1: SL-R algorithm

```
P = 4n × 4n identity matrix
foreach relative pose constraint do
    fill in the corresponding block of P with the constraint transformation
    fill in the transpose block with the inverse
while not completely solved do
    if at least one available triangle closing exists then
        perform triangle closing (direct rule)
    else if newly solved label exists then
        propagate label transformation to relative poses with that label (label rule)
    else
        foreach label do
            prospectively run AX = XB solver on all known relative poses with a label
        pick the most determined label and use that value for the label (AX = XB rule)
```

Properties of As	Ambiguities
At least two non-parallel rotations	Completely determined
Parallel rotations	Ambiguous translation along rotation axis
Any rotation	Ambiguous rotation and translation along rotation axis
No rotations, at least two non-parallel translations only	Ambiguous translation
No rotations, any translation	Ambiguous rotation along translation axis, ambiguous translation
No movement or no information	Completely ambiguous

Table 1 Classification of ambiguity of $AX = XB$ equations.

practice, the relative pose constraints will come from noisy measurements, and so it will not be possible to satisfy all constraints exactly. To account for this, we use local averaging to combine multiple inferences of a pose when more than one is available. Then, in the end, we perform nonlinear optimization afterwards to refine the result to a locally least-squares optimal solution in terms of reprojection error; this is described in Section 4.

3.2 SL-R algorithm

In this section we present an algorithm for SL-R. Our algorithm runs in polynomial time but is not guaranteed to produce a correct solution; as we will show in Appendix B, SL-R is NP-hard and so no known polynomial-time algorithm can produce a correct solution in all cases. However, as part of Caliber, our algorithm has worked in all practical solvable cases we have encountered so far, and is additionally able to detect and classify common forms of ambiguity.

3.2.1 General strategy and considerations

Recall from Section 3.1 that a solution to SL-R can be defined as an assignment of relative poses P_{ij} such that the composition of relative poses around any cycle is the identity, and all constraints are satisfied. This means that each cycle of poses corresponds exactly to an equation of rigid transformation matrices, with these equations being related to each other via the constraints. Our algorithm uses P_{ij} as the variables.

There are some special considerations that arise:

- Real measurements are noisy. If there are multiple ways of estimating a particular relative pose, each may produce a different result. Therefore, a way to average these estimates can improve the estimate.
- Even in the absence of noise, many instances of SL-R are inherently ambiguous; that is, there exists more than one solution. Fortunately, it is usually practically sufficient to select *a* solution in this case: if more than one solution fits the calibration data, this usually implies that they will make identical predictions as well. It is also useful to automatically detect and, where possible, classify these ambiguities.

Our algorithm incrementally builds a solution by applying a series of inference rules. Each of these rules produces an estimate of a relative pose by solving an available equation or system of equations. In order of highest to lowest priority, these are the direct rule, the label rule, and the $AX = XB$ rule. The algorithm is summarized in Algorithm 1.

3.2.2 Initial state

We represent the relative poses of SL-R as a block matrix P whose blocks are the relative poses P_{ij} represented as 4x4 rigid transformation matrices.

The condition that the composition of relative poses around any cycle must equal the identity has two basic implications: $P_{ii} = I$ (each absolute pose is identical to itself), and $P_{ij} = P_{ji}^{-1}$ (the relative pose from one absolute pose to another is the inverse of the relative pose in the other direction).

Therefore, we initialize P to contain I on its diagonal blocks and write the relative pose constraints into the matrix directly. Any block without a relative pose constraint is initially set to 0, indicating an unknown relative pose.

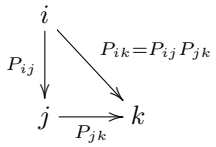
Here is an example with four absolute poses and a single relative pose constraint $P_{2,4} = T$:

$$P = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & T \\ 0 & 0 & I & 0 \\ 0 & T^{-1} & 0 & I \end{bmatrix} \tag{4}$$

3.2.3 Direct rule

The direct rule is derived from the constraint that the composition of transformations around any cycle should equal the identity. The simplest case is triangle-closing: if we know the values of two relative poses that both involve the same absolute pose, we can estimate the relative pose between the other two absolute poses as the composition of those two transformations.

A pictorial representation of this triangle-closing, where points represent absolute poses and arrows represent transformations:



(5)

The relative pose that closes a longer cycle can be estimated by applying this triangle-closing multiple times. Therefore, at the basic level, our direct rule performs this triangle-closing. A simple method of expressing triangle closing over all possible triples of absolute poses simultaneously is to use block matrix multiplication. Specifically, the block elements of the product $PP = P^2$ are:

$$\left(P^2\right)_{ik} = \sum_{j=1}^{n} P_{ij} P_{jk} \tag{6}$$

Each term in the sum is an estimate for the relative pose P_{ik} between absolute poses i and k iff both P_{ij} and P_{jk} are nonzero (that is, known), and zero otherwise. If the sum is zero, the relative pose could not be solved for via triangle closing and is therefore unknown. Otherwise, dividing by the homogeneous coordinate (bottom-rightmost element) gives us an average of several estimates, which we then project onto $SE(3)$ via SVD.

We apply the direct rule repeatedly until no new relative poses are estimated.

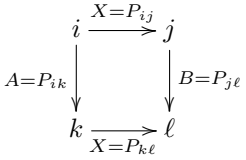
3.2.4 Label rule

If the direct rule is unable to make progress, we move on to the label rule, which is a direct application of the rigidity constraints. For each label, if at least one of the relative poses with that label has a known rigid transformation, we set all relative poses with that label to that rigid transformation. Like the direct rule, if there are multiple estimates for the same label, we take the average and project onto $SE(3)$.

3.2.5 $AX = XB$ rule

If neither of the previous rules can be applied, we use our final rule, which relies on solving systems of $AX = XB$ equations. It is this rule which allows our algorithm to solve novel and non-trivial problems.

For every pair $P_{ij}, P_{k\ell}$ of relative poses with the same unknown label X, we check if the relative poses $A = P_{ik}$ and $B = P_{j\ell}$ are both known. If so, the four relative poses form a cycle corresponding to an $AX = XB$ equation with A and B known and X unknown. Pictorially, such an equation looks like this:



(7)

Since labels can potentially appear many times in SL-R, there may be multiple such cycles for the same label X, each of which produces a different $AX = XB$ equation. Depending on the values of the As and Bs and the number of equations, the solution space for each label may have a different number of dimensions. In other words, some labels may be more constrained by their $AX = XB$ equations than others. We choose the most determined label to solve; if the solution is incompletely determined, we choose the solution that is closest to zero rotation angle and zero translation. In the presence of noise, we use a least-squares solution that accounts for all available $AX = XB$ equations for

each label, and when determining how many dimensions the solution space of a system of $AX = XB$ equations has, we have minimum tolerances for which we consider two vectors to be nonzero (translation) or nonparallel (rotation). We report the type of ambiguity of the chosen label, then return to the previous rules.

The specific cases of $AX = XB$ are discussed in Appendix A. A brief summary appears in Table 1.

3.2.6 Possible outcomes

Termination occurs in polynomial time: each outer loop iteration takes polynomial time, and since we can always guess a relative pose in the worst case, there are at most a polynomial number of iterations.

Fully determined solution. If our algorithm reports that the solution is completely determined, the solution is guaranteed to be unique and correct in the exact case. Practically, in the presence of noise, this produces a solution near the unique optimum.

Ambiguity. For some problems there may not exist a unique solution at all. However, this usually does not preclude a useful calibration. For example, a revolute joint (such as in our two-arm case—see Figure 4) produces rotations along only a single axis. In this case it is impossible to distinguish (from *e.g.* observations of a calibration target mounted on the joint) a difference in the location of the joint along the axis from a difference in the distance from the joint to the target along that axis. In this case, our algorithm would set one to zero and solve for the other. In this and all other practical cases we have encountered so far, our algorithm will succeed in finding *a* correct solution via the $AX = XB$ rule, and the arbitrary choices of our algorithm have no effect on the predictions made by the resulting model as long as the predictions do not exercise any degrees of freedom that were not present in the calibration data. Intuitively, an ambiguity may correspond to a parameter of the system that does not produce a visible effect in the observations. However, as long as future predictions are made under similar conditions, an arbitrary choice made to resolve that ambiguity will not be visible in those predictions either.

3.2.7 Complexity

As discussed in Appendix B it is possible to construct calibration instances that cannot be resolved by our $AX = XB$ analysis—indeed, that encode NP-complete problems. Given such an instance, our algorithm will likely guess a wrong value for a label. Again, we have not yet come across any practical calibration schemes where this occurs.

4 Caliber

The previous section discussed the core challenges in creating a general calibration system that solves the kinds of problems sketched in the introduction and proposed an algorithm for doing so. In this section we proceed to discuss Caliber, a calibration system for geometrically constrained camera systems that is built around that algorithm. By being able to handle rigidity constraints, Caliber provides greater generality than previously existing systems, and by using a final nonlinear optimization stage over all parameters, Caliber produces locally least-squares optimal solutions.

In order to make the system easy to use, the input to Caliber is in the form of a description of the system being calibrated together with the observations that were made on it. This is equivalent to an SL-R problem instance, but much easier for a user to specify; the system takes care of converting it to an SL-R problem internally.

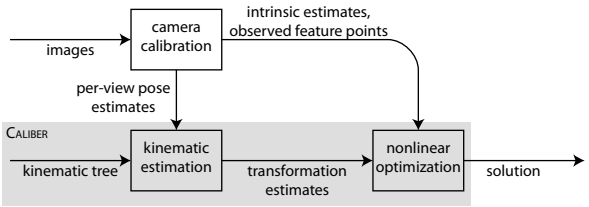


Fig. 3 The structure of Caliber: before Caliber is run, the images are processed by a standard **camera calibration** tool (Bouguet, 2010) to find image-space features, camera intrinsics, and per-image pose estimates. Next, our **kinematic estimation** phase computes approximate transformations according to a tree-structured model of the setup; finally, a **nonlinear optimization** refines this estimate to a locally least-squares optimal solution.

4.1 Overall structure

As illustrated in Figure 3, calibration using Caliber begins with a set of photographs of calibration targets from the camera(s) of the system. Both targets and cameras are attached to the system, and during calibration multiple photographs are taken as the system is taken through its range of motion. We then run Zhang's algorithm to determine the intrinsics of each camera and an estimated pose relating the camera to a calibration object in each view. These poses, together with a kinematic tree describing the geometric constraints,

are used in the second stage to estimate the underlying properties of the constrained system accounting for constraints such as rigid links and known motions. Finally, all the system parameters including camera intrinsics are jointly optimized to find a locally least-squares optimal solution.

Camera calibration. The physical calibration process consists of taking photographs from several views from camera(s) of calibration target(s), ideally getting a variety of angles relative to the calibration target(s) and exercising all of the degrees of freedom of the system. Given a set of views, the Zhang method is able to estimate the relative pose between the camera and target for each view, as well as the camera intrinsics. This can be repeated for each camera in the calibration setup.

Although Zhang's method is able to give us pose estimates by itself, these estimates are completely independent from view to view. It is unable to understand or estimate structural constraints on the geometry of the calibration setup such as the separation between the cameras of a rigid camera cluster, or the location of the center of a joint that a camera is mounted on.

Kinematic estimation. To solve this, we have a kinematic estimation stage which incorporates rigidity constraints using our SL-R algorithm. Given a description of the system as a kinematic tree, and the estimates from the camera calibration stage, our kinematic estimation determines an estimate for each label transformation automatically.

Nonlinear optimization. The result of our kinematic estimation may not be optimal in terms of reprojection error across all images. Our final stage takes the estimates from the kinematic estimation and the intrinsic estimates from the camera calibration. It performs nonlinear optimization over the relative poses in the kinematic tree and any intrinsic parameters to produce a locally least-squares optimal solution.

4.2 CALIBER problem definition

Although SL-R is a useful representation from an theoretical point of view, it has two shortcomings from a usability point of view. First, it can be cumbersome and unintuitive for a user to directly specify a complete instance of SL-R and interpret the solution. Second, SL-R does not define an objective function and parameterization on which to perform nonlinear optimization.

To address these, we observe that the inherent geometric structure of a broad category of camera systems can be described using a **kinematic tree**, akin to a scene graph, that expresses the relationships between objects (*e.g.* cameras, calibration targets, robot joints) in terms of nested frames of reference. See Figure 4 for examples. The user of CALIBER describes the calibration problem by specifying this kinematic tree, indicating which transformations in the tree are known and which need to be solved for, and providing a set of camera observations that link nodes containing cameras to other nodes containing objects observed by the cameras. This description is equivalent in power to the SL-R representation, but is more intuitive for a user to specify, and defines a desired parameterization for the optimization stage. As we shall see in Section 4.4, this representation also provides an advantage in creating instances of SL-R that are solvable by our algorithm.

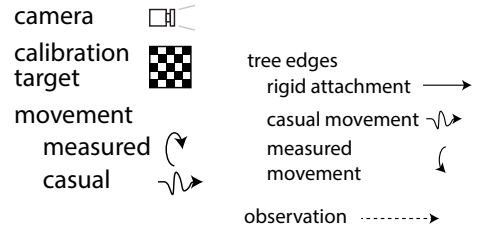
4.2.1 Components

The kinematic tree is defined by the following components:

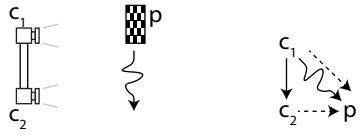
Nodes. Nodes represent objects in the system—cameras, calibration targets, the two sides of a robot arm joint, and so forth. Each node has exactly one parent, except for a single root node which represents the global, "laboratory" frame. The nodes thus form a tree. Each node can also hold any other variables, such as intrinsic parameters, that are not used in the kinematic estimation stage but will be optimized over in the nonlinear optimization stage.

Observations. In addition to nodes, we have observations. An observation is defined by the following:

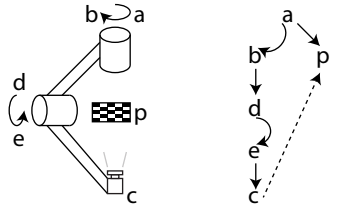
- A source node (representing a camera) and a target node (representing a calibration target).
- An estimate of the relative pose between the source and target node. This produces a relative pose constraint.
- The state of the system at the time the observation was taken. Specifically, this is defined by relative poses between each pair of adjacent nodes on the paths from the root to the source and target nodes. Each of these can be initially known, in which case they are specified as a rigid transformation value, or initially unknown, in which case they get a specified label as in the rigidity constraints of SL-R. Labels can be reused across different observations, resulting in shared rigidity constraints. For example, if one observation labels a relative pose "A", and another observation labels also labels a relative pose "A", then those two relative poses are constrained to have the same value.



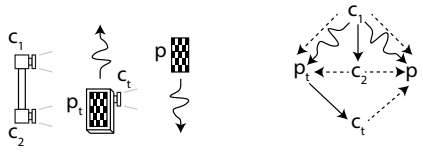
(a) **Legend.**



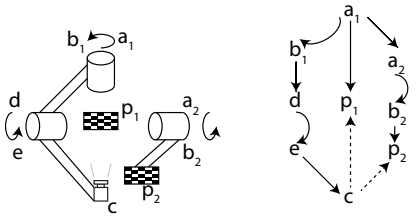
(b) **Stereo** pair.



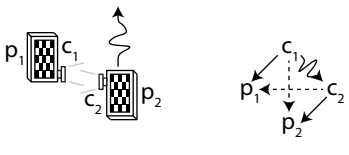
(c) Spherical **gantry.** The camera arm has two revolute joints with different axes of rotation.



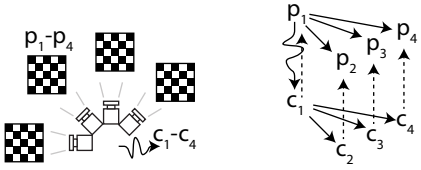
(d) Stereo pair with **tablet.** The goal is to calibrate the camera and display on a tablet computer.



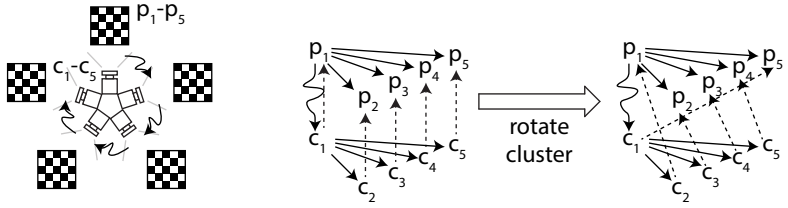
(e) **Two-arm** spherical gantry. The second arm has only a single axis of rotation.



(f) **Facing** smartphones. These are two devices with screens and front-facing cameras that take photographs of each other.



(g) **Ad-hoc** cluster.



(h) Point Grey **Ladybug.** The symmetrical arrangement of the cameras allows us to rotate the cluster so that each camera sees the next target in turn. Simulation suggests this improves the calibration. Here we have drawn the same tree to show the change in the source and target nodes of the observations after such a rotation.

Fig. 4 Calibration setups and their corresponding trees. Drawings not to scale.

– The reprojection error function of that view, defined in terms of the relative pose between its source and target nodes and any variables such as intrinsic parameters. See 4.5 for details.

The kinematic estimation stage constructs and solves an instance of SL-R using the relative pose information provided in the observations. Then, the nonlinear optimization stage optimizes over the aggregate of the reprojection errors for all observations.

Note that although Caliber uses cameras, it is not bound to them: any measurement device that can produce a relative pose estimate would work. Likewise, reprojection error is just one option for an objective function; any objective function defined in terms of the relative pose between the source and target nodes and some intrinsic parameters could be used.

4.3 Camera calibration via Zhang's method

Caliber uses Zhang's algorithm, via Bouguet's (2010) implementation, to get rough estimates of relative poses between cameras and targets and of the camera intrinsics. Bouguet takes a set of photographs of a checkerboard calibration target from a single camera, determines the 2D position of grid intersections in each, and uses these to estimate the relative pose between the camera and the target in each photograph, as well as a single set of camera intrinsics for the entire set. Therefore, we run Bouguet over all of the photographs that came from each camera to produce a relative pose estimate between the camera and calibration target for each view, and intrinsic parameter estimates for each camera.

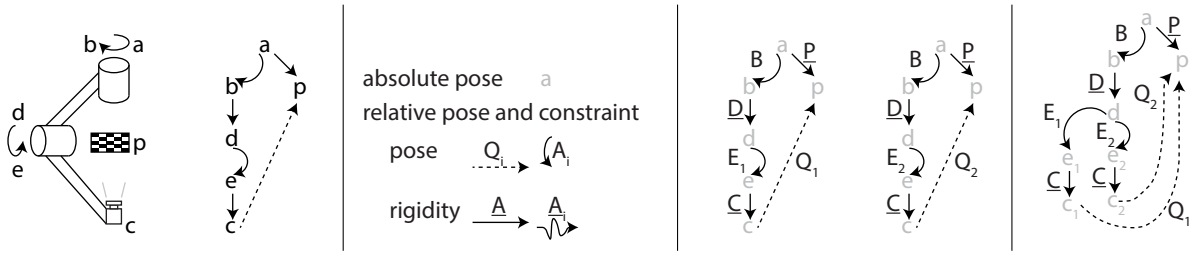


Fig. 5 Reduction from the kinematic tree to SL-R for the spherical gantry case of Figure 4 (c). **Far left.** Reproduction of the setup diagram and kinematic tree. **Center left.** Legend. Each node in the kinematic tree may produce more than one corresponding absolute pose. After the reduction, all edges become relative poses. Measured relative poses get a relative pose constraint, and non-measured relative poses get a rigidity constraint (underlined). **Center right.** Two observations of the spherical gantry. Each consists of a path from the root to a source and to a target node, and a cross-edge between the source and target nodes. In this example, the top joint transformation B remains the same but the bottom joint transformation E changes. **Far right.** The corresponding SL-R instance. The top part of both cycles map onto the same absolute poses since they are the same, whereas the differing transformation for E produces two absolute poses for each node below it. Observe that the cycle consisting of E_1, E_2, Q_1, Q_2, and the $\underline{C}$s corresponds to an equation of the form $AX = XB$.

Although Bouguet's implementation is a useful building block, CALIBER is not tied to this particular implementation; it is simply a convenient way of generating relative pose estimates.

4.4 Kinematic estimation

The goal of the kinematic estimation is to produce an estimate for each label that is consistent with all of the observations. Our kinematic estimation stage reduces the kinematic tree to SL-R and solves it using our SL-R algorithm.

We perform our reduction according to the following correspondence between the kinematic tree and an instance of SL-R:

- Each appearance of a node in an observation in the kinematic tree maps to an absolute pose in SL-R—though as we will explain shortly, multiple appearances of the same node may map to the same absolute pose.
- Each known or estimated relative pose (dotted "observation" and curved "measured movement" arrows in Figure 4) mentioned in an observation, including the estimated transformation between the source and target node, becomes a relative pose constraint in SL-R between the corresponding absolute poses.
- Each rigid unknown relative pose (straight solid "rigid attachment" arrows in Figure 4) between two nodes becomes a rigidity constraint between the corresponding absolute poses.
- We need not include the squiggly "casual movement" arrows in Figure 4 as their relative pose is not held fixed between observations.

To keep the number of nodes to a minimum and give our SL-R algorithm information about which absolute poses are identical, we use the structure of the kinematic tree to guide the construction of our instance of SL-R. The core idea behind our strategy is to consider each pose in the frame of the root node, and make sure that symbolically equivalent poses are assigned the same absolute pose in our instance of SL-R. To do this, we keep a dictionary of which absolute poses we have seen and reuse them if they are involved in multiple observations. We then connect each absolute poses using the constraints specified in the observations involving that absolute pose. Pseudocode is shown in Algorithm 2, and a pictorial example is shown in Figure 5.

4.5 Nonlinear optimization

Once the kinematic estimation is done, we have a complete set of estimates for the rigidity constraints. From here, given a desired parameterization of individual transformations in the tree, we can automatically compute the reprojection error across all images and its Jacobian with respect to those parameters and the intrinsic parameters of the cameras. We can then apply a nonlinear optimization method such as Levenberg–Marquardt to refine the answer. This is similar to bundle adjustment, but respecting the specified kinematic tree and its implied rigidity constraints. Note that the transformations initially estimated by the kinematic estimation stage are *not* held fixed during optimization: the optimization is allowed to jointly refine these transformations (subject to rigidity constraints) as well any other parameters such as camera intrinsics. The solution that minimizes least-squares reprojection error is statistically optimal, in the sense that it is a maximum likelihood estimate of

Algorithm 2: Reduction to SL-R

```
absolutePoses = new dictionary: sequence of constraints from root to current node → absolute poses
foreach observation do
    absolute source pose = processPath(observation source)
    absolute target pose = processPath(observation target)
    add relative pose constraint between the absolute source and target poses with the observation's pose estimate
Function processPath(stNode)
    foreach node and relative pose on path from root to stNode do
        current absolute pose = absolutePoses[sequence of constraints from root to current node]
            (create new absolute pose if not already existing)
        if relative pose is known then
            add relative pose constraint between current absolute pose and its parent
        else
            add rigidity constraint between current absolute pose and its parent
    return currentAbsolutePose
```

the parameters under the standard model of Gaussian noise in image space.

4.5.1 Objective function

The global objective function of the nonlinear optimization stage is the sum-of-squares of the reprojection error across all observations (defined in Section 4.2.1).

For each observation i, the reprojection error is the difference between the image points predicted by the projection function f_i, and the actually observed image points $\mathbf{y}_i$:

$$\mathbf{z}_i = f_i\left(p_1, p_2, \ldots, \mathbf{x}_i\right) - \mathbf{y}_i \tag{8}$$

where $\mathbf{x}_i$ is the coordinates of the calibration target's feature points (in the target frame) expressed as a matrix of column vectors and $p_1, p_2, \ldots$ are the parameters into the projection function. The nonlinear optimization minimizes the sum of squares of these $\mathbf{z}_i$ across all observed points and all observations.

The projection function can be further decomposed into two parts. The intrinsic part computes the predicted image points given a set of intrinsic parameters such as focal length, principal point, and radial distortion, and the coordinates of the target points in the camera's frame. The coordinates of the target points in the camera's frame are equal to their coordinates in the target frame $\mathbf{x}_i$, transformed by the relative pose P_{st} between the camera and the target. The other part of the projection function computes this relative pose between camera and target P_{st} as a function of extrinsic parameters such as rotations and translations of relative poses in the kinematic tree.

$$f_i = f_{i,\text{int.}}\left(p_{\text{int.},1}, p_{\text{int.},2}, \ldots, P_{st}\mathbf{x}_i\right) \tag{9}$$

$$P_{st} = f_{i,\text{ext.}}\left(p_{\text{ext.},1}, p_{\text{ext.},2}, \ldots\right) \tag{10}$$

We will now discuss the specific parameters that we use.

4.5.2 Parameterization

The intrinsics can be parameterized as per the standard projection function of *e.g.* (Hartley and Zisserman, 2004), whose parameters are focal length, principal point, and radial distortion.

As for the extrinsics, we parameterize the label transformations using three rotation and three translational parameters. The translation parameters are simply the three Cartesian coordinates of the translation. We use the axis-angle representation for rotations and parameterize the rotations as a second rotation relative to the rotation produced by the kinematic estimation. Symbolically, given an initial rotation estimate R_{KE} and a rotation parameter vector r, the net rotation is

$$R_{\text{net}} = R_{\text{KE}} e^{[r]_\times} \tag{11}$$

where $[r]_\times$ is the cross-product matrix corresponding to r and $e^{[r]_\times}$ is the axis-angle rotation implied by r. Since we expect the kinematic estimation to produce a good estimate for the rotation, we expect this second rotation to stay close to the identity throughout the optimization process.

Note that the full 6-D parameterization for each label transformation may be redundant in some cases. Here we rely on the regularization of nonlinear optimizers such as Levenberg-Marquardt to keep the solution near the initial guess along the ambiguous degrees of freedom.

4.5.3 Jacobian

To perform nonlinear optimization efficently, we need to compute the Jacobian of the reprojection error. The

Jacobian of the intrinsic parameters is straightforward to analyze and will not be covered here. This leaves the Jacobian of the extrinsics. Given an observation with camera node s and calibration target node t, the relative pose between them P_{st} is $P_s^{-1}P_t$. In turn, P_s is the composition of the poses on the path from the root to s, and likewise for t:

$$P_s = \prod_{S_i \text{ on path from root to } s} S_i \tag{12}$$

$$P_t = \prod_{S_i \text{ on path from root to } t} S_i \tag{13}$$

For any extrinsic parameter p we can determine $\frac{dP_s}{dp}$ and $\frac{dP_t}{dp}$ via the product rule. Then

$$\frac{dP_{st}}{dp} = \frac{d}{dp}\left(P_s^{-1}P_t\right) \tag{14}$$

$$= \frac{dP_s^{-1}}{dp}P_t + P_s^{-1}\frac{dP_t}{dp} \tag{15}$$

$$= -P_s^{-1}\frac{dP_s}{dp}P_s^{-1}P_t + P_s^{-1}\frac{dP_t}{dp} \tag{16}$$

gives us the derivative of the pose between the camera and the target P_{st} with respect to the extrinsic parameter p.

5 Evaluation and results

5.1 Evaluation

In evaluating the results of experiments, we need to distinguish between evaluating CALIBER itself and evaluating particular calibration setups.

5.1.1 Evaluating CALIBER

We wish to evaluate CALIBER based on whether the locally least-squares optimal solution it produces is the globally optimal solution. Unfortunately, we know of no feasible method of proving whether a particular locally optimal solution is globally optimal. And while our ultimate goal is to be able to calibrate real systems, we cannot simply take a real system and compare CALIBER's result to ground truth either: there is a chicken-and-egg problem in that the calibration system we are evaluating is itself the best, or even only, method of trying to determine the ground truth in the first place. Many of our setups were constructed by hand without *a priori* measurements, and even in setups involving purpose-built equipment, such as the two-arm spherical gantry, we found that the calibration can outperform construction specifications.

Therefore, to evaluate the result of CALIBER compared to the globally optimal solution, we turn to synthetic data. We took CALIBER's solution for each of our real experiments in 5.2, and used it as ground truth for a synthetic counterpart. Specifically, we reprojected all of the image points using the solution, added Gaussian noise with standard deviation matched to the residual of the Zhang calibration stage, and then used this synthetic data as input of a new calibration problem. The ground truth solution therefore has RMS reprojection error approximately equal to the standard deviation of the noise added. Then we reinitialized and ran CALIBER on this new, synthetic calibration problem. In order to give a realistic amount of deviation of the initial pose estimates from ground truth, the reinitialization used the original Bouguet pose estimates rather the optimized pose estimates from the original solution. We expect the globally optimal solution to be close to the ground truth, and thus for it to have a similar RMS reprojection error as well. If our kinematic estimation is good, we expect our locally optimal solution to be close to the globally optimal solution, and, if the kinematic estimation falls within the basin of convergence of the global optimum, the locally optimal solution will also be the globally optimal solution.

In each case, the RMS reprojection error of CALIBER's solution matched the magnitude of the added noise to within three percent. Therefore, here the local solution produced by CALIBER is either globally optimal or at least statistically resembles such.

5.1.2 Evaluating a calibration scheme

Even if CALIBER does find the globally optimal solution, this does not mean that all calibration setups produce equally good results. Now we must define what constitutes a "good" result.

Problems with evaluating parameter values directly. One obvious possible definition of "good" is how close the solved parameter values are to ground truth. However, such an approach has the same problem of determining ground truth as discussed before.

It is also unclear what is the appropriate baseline to compare such errors to. For example, even within a single experiment the magnitude of the translational errors is subject to choice of units of measurement. Comparison of errors in parameter values between different setups is even more problematic; the parameters in one setup may not have the same interactions, the same units, or even the same dimensionality as those of another.

Cross-validation. Instead, we define "practically relevant" in terms of the predictive power of the results of Caliber. This has several advantages. First, it uses the same metric that Caliber is optimizing for, namely reprojection error. Second, the ground truth of this definition is something that we directly measured, namely detected image points. Finally, reprojection error is always measured in pixels, allowing an "apples-to-apples" comparison between the results of different setups.

To evaluate the predictive power of Caliber, we used leave-one-out cross-validation (LOOCV) for both synthetic data (as above) and experimental data. This operated on each experiment as follows: for each view, we ran one trial of Caliber with that view omitted, predicted the image points for the omitted view, and recorded the errors in the predicted image coordinates. We then aggregated these errors across all trials to produce our **LOOCV prediction errors**.

In interpreting the results of this cross-validation, we need to distinguish between different components of this prediction error:

- **Random error.** This results from noise in the feature points of the view to be predicted. This component of the error would still exist even if the parameter values produced by Caliber were perfect.
- **Parameter error.** The feature points we used in each trial to calibrate the system are also noisy. This noise propagates into errors in the parameter values produced by Caliber.
- **Systematic error.** Finally, our intrinsic and extrinsic models are imperfect: camera lenses may not fit our projection model exactly, robot arm links may flex, joints may have hysteresis and backlash, and so forth. Such weaknesses in our models produce systematic error.

We did not observe any obvious structure in the Zhang-only reprojection error; thus, we consider the systematic error in the Zhang result to be negligible. Since the Zhang model has a separate pose for each view, and is thus over-parameterized relative to the kinematic tree, we consider the parameter error in the Zhang result to be negligible as well. This leaves the random error in the feature points, which we estimate as the Zhang method's reprojection error over all images (*i.e.* not leaving one out). The synthetic data is generated as if there was no systematic error; therefore, we use the LOOCV prediction error for synthetic data as an estimate for the random error plus parameter error. Finally, we estimate the total error as the LOOCV prediction error for experimental data. The flow of data in our evaluation is summarized in Figure 7.

The machinery for both generating synthetic test data from an existing calibration and performing LOOCV is integrated into Caliber itself, allowing it to be a tool for evaluating calibration schemes in addition to its primary use in performing the calibrations themselves.

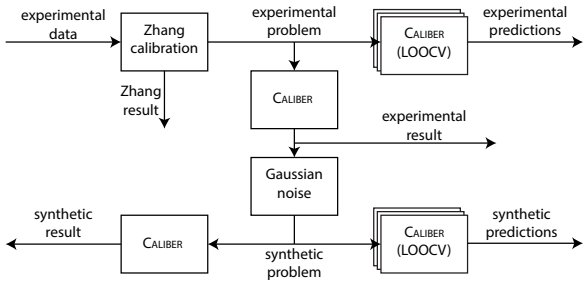


Fig. 7 Flow of our evaluation scheme.

5.2 Experiments and results

Our experimental setups are depicted in Figure 4 and results are summarized in Figure 8. We ran each kinematic estimation and nonlinear optimization on an Intel i5-760 processor with 4 cores. Each calibration took less than ten seconds to compute. A brief discussion of each setup and its results follows. Example images for the two-arm setup from the calibration process are shown in Figure 6. Additional details, results, and images for each setup can be found in the supplemental material.

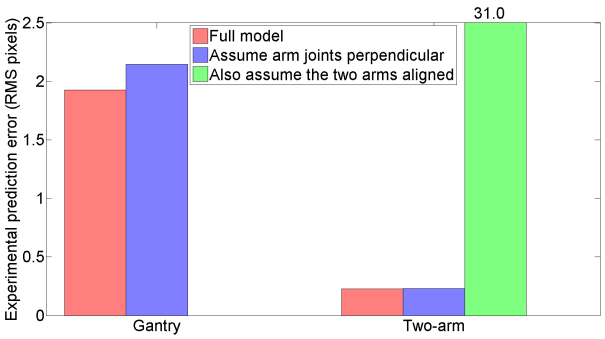


Fig. 9 Comparison of different kinematic models for the gantry and two-arm cases. All of the models had the same prediction error for the synthetic data to within 3%. However, the prediction errors for the experimental data show that allowing Caliber to solve for the relative pose between the two gantry arms produces a large improvement over assuming they are aligned, and allowing it to solve for the relative pose between the two camera arm joints produces a modest but noticeable improvement over assuming they are perpendicular.

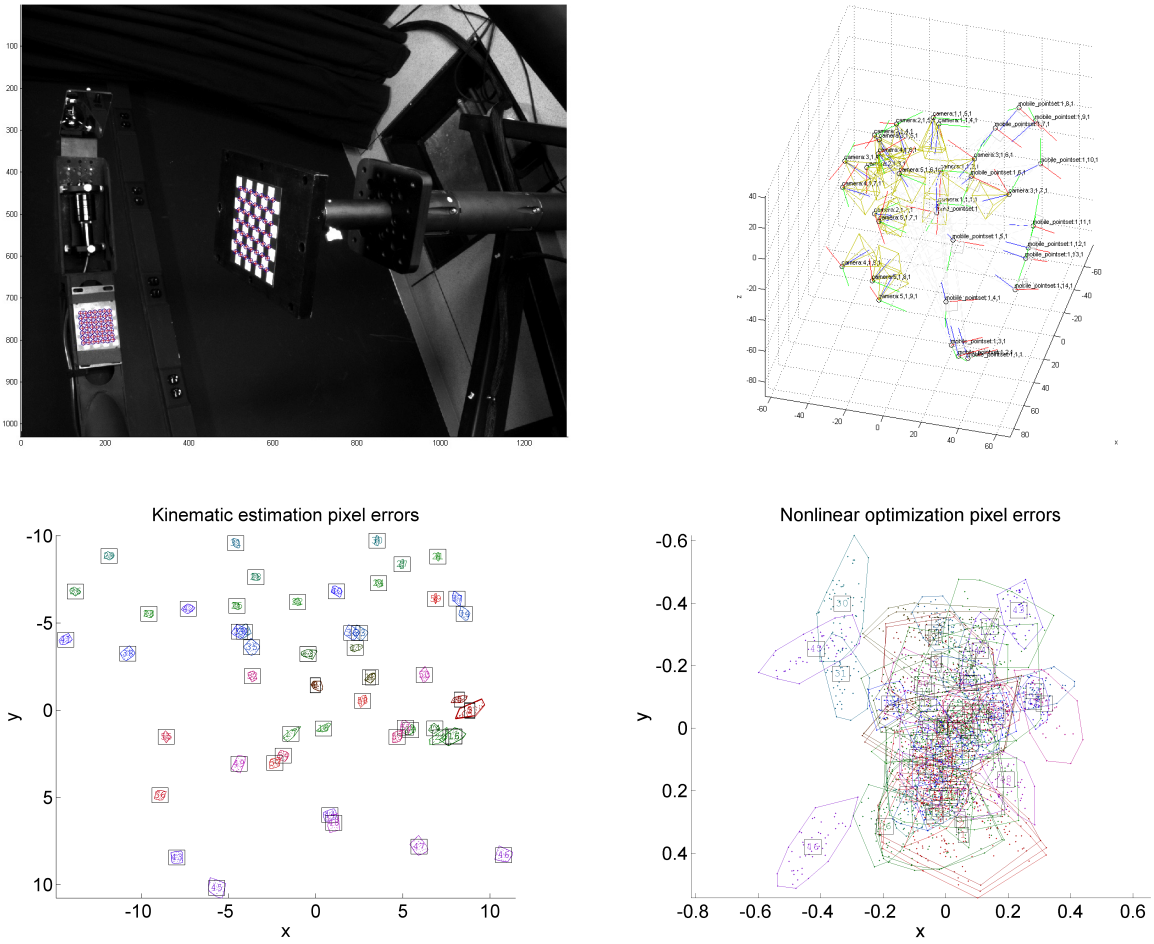


Fig. 6 Example images from the calibration process for the two-arm case. See the supplemental material for more images of this and the other setups. **Top left:** Example input image. The blue circles indicate observed checkerboard intersections. The red crosses indicate reprojections of the complete optimization. **Top right:** A depiction of the camera and calibration target frames for each observation. The x, y, and z axes are red, green, and blue respectively. Axes are labeled with the observation they came from. Origins are the camera center for cameras and one corner of the checkerboard for calibration targets. **Bottom:** Pixel residuals after kinematic estimation (but before nonlinear optimization) and after nonlinear optimization. Colored pixels represent residuals for a single pixel of the correspondingly-colored observation. Colored polygons represent convex hulls for the residuals of the correspondingly-colored observation. There is one numbered box for each observation, at the average pixel residual for that observation. Note the change in scale between plots.

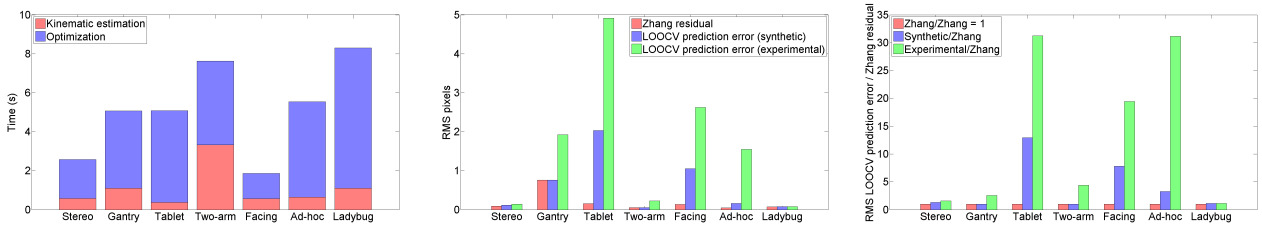


Fig. 8 Results. **Left:** Kinematic estimation and optimization running time. **Center:** RMS residual for LOOCV prediction errors for synthetic and experimental data compared to the Zhang model by itself. Note that the Zhang model by itself uses independent poses for each observation and is thus strictly less constrained than our model. In particular, it does not obey pose or rigidity constraints and thus is non-predictive with respect to poses. Therefore, its residual is necessarily equal to or less than any model which *does* have to obey those constraints (such as ours). We include it here only as a baseline. **Right:** As center, but normalized by Zhang RMS residuals.

5.2.1 Stereo pair

This is probably the simplest possible case that does not reduce to the basic Zhang method, and Bouguet has special-purpose code for this case, which provides a reference implementation to compare against. This made this a good sanity check for Caliber.

We were able to duplicate the results of the Bouguet (2010) calibration toolbox's special-case code for stereo cameras to within a margin that could be attributed to machine precision, and the calibration performed well overall, with both experimental and synthetic prediction errors being only a modest factor larger than the Zhang residual.

5.2.2 Spherical gantry

This was the case that originally motivated this project. The gantry allows one to position a camera at any angle relative to an object to be photographed with high *precision*. However, calibration is required to make these angles highly *accurate* as well, which is important for applications such as reflectance measurement.

At first we assumed the two joints of the camera arm were indeed perfectly perpendicular. However, we found that allowing Caliber to solve for the relative pose between the two joints resulted in better predictions. The improvement was more obvious in the two-arm case, which used the same gantry. See Figure 9 for a comparison. This calibration showed an unusually high error in the Zhang residual (*i.e.* what appears to be random error), even compared to the two-arm case, but performed well overall. This may have been due to the camera we were using, which was different in the two cases.

5.2.3 Stereo pair with tablet computer

At its core, this setup attempted to measure the separation between a tablet computer's screen and camera. The tablet cannot see its own screen, so we viewed the tablet with a stereo pair, and added another calibration target that could be seen by all three cameras.

This was the most physically challenging experiment to perform due to the difficulty of holding the tablet in place. The calibration was the worst of our cases overall in both experimental and synthetic data, with the views from the tablet computer being the worst within this setup, possibly because it could only be predicted using two other observations in serial.

5.2.4 Two-arm spherical gantry

For the reflectance measurement application, it is often useful to be able to change the lighting direction in addition to the viewing direction. This can be accomplished by mounting a light source on a second arm. Of course, this means that the second arm must also be calibrated to produce accurate angles.

As with the simpler gantry case, we at first assumed the two joints of the camera arm were indeed perfectly perpendicular, and furthermore that the axis of rotation of the second arm was aligned with the first. However, we again found that allowing Caliber to solve for the transformation between the two joints resulted in better predictions, as well as the transformation between the axes of the two arms. See Figure 9 again for a comparison.

The reprojection error for the solution does show some obvious structure, but in absolute terms the predictions for this setup were quite good for both synthetic and experimental data, suggesting that this setup is well-conditioned. More detailed plots of the reprojection error pattern as well as comparisons between the reprojection error for different models can be found in the supplemental material.

5.2.5 Facing smartphones

We wanted to see if we could calibrate two smartphones just by having them taking simultaneous pictures of each other. This type of calibration is appealing because it would not require any objects other than the devices to be calibrated. We used laptops instead of true smartphones since it was easier to set up a system for taking photographs without having to directly manipulate the devices.

Another purpose of this experiment was to demonstrate a practical, solvable case that could not be analyzed using our $AX = XB$ rule, but to our surprise Caliber determined that it actually was a case of $AX = XB$. However, the resulting calibration was one of the weaker ones in our set with relatively high prediction errors for both synthetic and experimental data, and more numerically stable setups might be sought.

5.2.6 Ad-hoc cluster

Calibrating clusters of non-overlapping cameras is useful for applications such as multiple cameras mounted rigidly on an autonomous vehicle. To model this, we bolted four cameras, including two DSLRs, onto a bar, mounted the cluster on a tripod, and placed a calibration target roughly in front of each camera.

This setup seemed to particularly suffer from systematic error. Given the weight of the DSLRs and only a single mounting point per camera, it seems likely this was due to the setup being less rigid than our kinematic model implied.

5.2.7 Point Grey Ladybug

We also tried a purpose-built cluster, the Point Grey Ladybug, which has featured in previous non-overlapping camera calibration work such as (Kumar et al, 2008) (though that particular work used the Ladybug2 as opposed to our Ladybug3). We only calibrated the five side cameras, but the sixth/top camera could be included by mounting a calibration target in front of it.

The fields of view of the cameras do actually overlap slightly; however, in order to demonstrate that CALIBER works in true non-overlapping cases, we did not take advantage of this fact. Indeed, including overlapping images actually *worsened* our cross-validation error. We believe this is because the photographs are highly distorted near the edges and corners, and the polynomial distortion model that we used does not handle this well, especially with a limited number of photographs. The edges and corners are exactly the parts of the photographs where overlapping views tend to occur.

The calibration of this setup performed much better than the facing and ad-hoc cases (the other non-overlapping or non-overlapping-like cases). We identify the following key advantages:

- We took a greater number of views with the Ladybug setup—more than twice as many as either of these other two cases.
- The Ladybug probably had better physical rigidity.
- The Ladybug setup had a greater number of cameras. Simulation (that is, constructing a synthetic scenario, adding noise, and running it through CALIBER) suggests that increasing the number of cameras in a cluster improves the calibration, even if the fields of view of the cameras remain non-overlapping and the number of views from each camera remains the same.
- The symmetrical arrangement of the cameras allowed us to rotate the unit so that each camera saw the next calibration target in turn. Again, simulation suggests that this improves the calibration even if no two cameras ever see the same calibration target at the same time and no camera ever sees more than one calibration target at the same time.

6 Conclusion

In this article we presented the sensor localization with rigidity problem (SL-R), which represents a class of localization problems involving not only relative pose constraints, but also rigidity constraints. Rigidity constraints encode the notion that the relative pose between two objects remains the same over time. We have shown that problem is NP-hard and gave an algorithm that is not guaranteed to produce a correct solution, but does so in practical cases, and runs in polynomial time.

Around our SL-R algorithm, we built CALIBER, a practical calibration system for kinematically constrained camera systems that presents an easy-to-use kinematic tree representation to the user and refines the solution produced by our SL-R algorithm to a locally least-squares optimal result in terms of reprojection error. We demonstrated CALIBER on a variety of both real and synthetic experimental data.

We will be releasing code to CALIBER, and hope that it will be found useful both as a standalone tool and as a base for adaptations and expansions.

6.1 Future work

Possible extensions to this work include the following:

Reflections. Some existing methods, *e.g.* (Kumar et al, 2008) use planar mirrors in their calibration systems. This could be a useful generalization of our system.

SL-R algorithm. We have found the efficiency and accuracy of our SL-R algorithm sufficient in practice. However, more efficient and accurate implementations may be possible. Our matrix multiplication-based implementation of the direct rule is the bottleneck for our current method, requiring a quadratic amount of storage and a n-by-n matrix multiplication for each of up to a linear number of outer loop iterations. This might be improved by collapsing connected components of known relative poses as the algorithm runs. We could also consider (Govindu, 2004)'s Lie algebraic averaging in order to improve the accuracy.

$AX = XB$ in the presence of noise. A more comprehensive analysis of $AX = XB$ in the presence of noise could lead to better accuracy and better decisions by our SL-R algorithm. For example, we might replace our discrete $AX = XB$ cases with a "soft" regularization that produces a finer-grained priority for which label to solve next.

Input recommendation. We could imagine an algorithm that recommends additional observations that may result in a better calibration. A sufficiently good such algorithm might even be used to automate the physical calibration process itself by directly controlling which observations are taken.

Input simplification. While our system frees the user from working out a complete calibration algorithm, it does require the user to input a description of the calibration setup's layout. Calibration could be made easier if this could be determined semi- or fully- automatically.

Acknowledgements We would like to thank Wenzel Jakob, Pramook Khungurn, Bruce Walter, and Rundong Wu for testing out Caliber on real problems and providing feedback.

References

Bouguet JY (2010) Camera calibration toolbox for Matlab

Chen H (1991) A screw motion approach to uniqueness analysis of head-eye geometry. In: CVPR, pp 145–151

Dieudonné Y, Labbani-Igbida O, Petit F (2010) Deterministic robot-network localization is hard. Trans Rob 26(2):331–339, DOI 10.1109/TRO.2010.2042753, URL http://dx.doi.org/10.1109/TRO.2010.2042753

Esquivel S, Woelk F, Koch R (2007) Calibration of a multi-camera rig from non-overlapping views. In: Hamprecht F, Schnörr C, Jähne B (eds) Pattern Recognition, Lecture Notes in Computer Science, vol 4713, Springer Berlin / Heidelberg, pp 82–91

Estrada C, Neira J, Tardós JD (2009) Finding good cycle constraints for large scale multi-robot slam. In: Proceedings of the 2009 IEEE international conference on Robotics and Automation, IEEE Press, Piscataway, NJ, USA, ICRA'09, pp 2427–2431, URL http://dl.acm.org/citation.cfm?id=1703775.1703843

Govindu V (2004) Lie-algebraic averaging for globally consistent motion estimation. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, vol 1, pp I–684–I–691 Vol.1, DOI 10.1109/CVPR.2004.1315098

Govindu VM (2001) Combining two-view constraints for motion estimation. In: In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp 218–225

Hartley RI, Zisserman A (2004) Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521623049

Heath G (2008) How to solve the matrix equation xax=b. URL http://www.mathworks.com/matlabcentral/newsreader/view_thread/165797

Kumar R, Ilie A, Frahm JM, Pollefeys M (2008) Simple calibration of non-overlapping cameras with a mirror. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp 1–7, DOI 10.1109/CVPR.2008.4587676

Kümmerle R, Grisetti G, Burgard W (2011) Simultaneous calibration, localization, and mapping. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, pp 3716–3721

Maybank SJ, Faugeras OD (1992) A theory of self-calibration of a moving camera. IJCV 8(2):123–151

Nister D, Kahl F, Stewenius H (2007) Structure from motion with missing data is np-hard. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pp 1–7, DOI 10.1109/ICCV.2007.4409095

Park F, Martin B (1994) Robot sensor calibration: solving $AX = XB$ on the euclidean group. IEEE Trans on Robotics and Automation 10(5):717 –721, DOI 10.1109/70.326576

Shiu YC, Ahmad S (1989) Calibration of wrist-mounted robotic sensors by solving homogeneoustransform equations of the form $AX = XB$. IEEE Trans on Robotics and Automation

Strobl KH, Hirzinger G (2006) Optimal hand-eye calibration. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 4647–4653

Thrun S, Montemerlo M (2005) The GraphSLAM algorithm with applications to large-scale mapping of urban structures. International Journal on Robotics Research 25(5/6):403–430

Tsai R, Lenz R (1989) A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. IEEE Trans on Robotics and Automation 5(3):345 –358, DOI 10.1109/70.34770

Tsai RY (1987) A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. IEEE Trans on Robotics and Automation 3(4):323–344

Zach C, Klopschitz M, Pollefeys M (2010) Disambiguating visual relations using loop constraints. In: CVPR, IEEE, pp 1426–1433, URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#ZachKP10

Zhang Z (1999) Flexible camera calibration by viewing a plane from unknown orientations. In: ICCV,

pp 666–673

A Systems of AX = XB equations

In this section we cover solutions to systems of $AX = XB$ equations of rigid transformations. These solutions are not new in an of themselves, but they are a subroutine that allows our algorithm to solve non-trivial instances of SL-R.

A.1 Overview

The problem of solving a system of $AX = XB$ equations of rigid transformations is defined as follows: given a set of pairs of 4x4 rigid transformation matrices A_i, B_i, determine a 4x4 rigid transformation X such that for all i, $A_iX = XB_i$. Systems of equations of this sort naturally arise from situations such as hand-eye calibration and non-overlapping views from rigidly attached cameras.

A.2 Separation of rotation and translation

We opt to separate the $AX = XB$ equation on full rigid transformations into its translational and rotational parts. This gives

$$R_A R_X = R_X R_B \tag{17}$$

$$(R_A - I)\,\mathbf{t}_x = R_X \mathbf{t}_b - \mathbf{t}_a \tag{18}$$

A.3 Screw theory

It is also useful to observe that these equations describe a similarity transformation $B = X^{-1}AX$. Furthermore, (Chen, 1991) gives a method of characterizing $AX = XB$ systems using screw theory. Briefly, screws are a way of representing rigid transformations. They consist of a screw axis, which is a line in space, a translation magnitude along that axis, and a rotation magnitude about that axis. Screws have the property that the translation and rotation magnitudes are invariant with rigid transformations, whereas the screw axis transforms with rigid transformations. Therefore, if we have one or more $AX = XB$ equations, we can think of this problem as one where we attempt to find the transformation X that takes the screw axes given by $A_1, A_2, \ldots$ to those given by $B_1, B_2, \ldots$

A.4 Cases with rotation

Nonparallel (skew or intersecting) screw axes (0 undetermined dimensions). (Park and Martin, 1994) gives a method to compute the unique answer in the case where the screw axes are nonparallel and the rotation magnitudes are not near the identity or a 180-degree rotation (so the screw axes are well-defined). If the problem is overconstrained by having more than two $AX = XB$ equations, they give a least-squares solution. In particular, they compute the rotational matrix that transforms a $3 \times n$ matrix of screw axis direction vectors α (corresponding to the As) to another one β (corresponding to the Bs) as

$$M = \beta\alpha^T \tag{19}$$

$$R = \left(M^T M\right)^{-\frac{1}{2}} M^T \tag{20}$$

After the rotation is solved for, they find a least-squares solution for the translation. We use this solution directly for the fully-determined case, and our solutions for the other cases are based on this method.

Parallel screw axes (1 undetermined dimension). (Chen, 1991) shows that in the case where the screw axes are parallel but not coincident, the rotation can be uniquely determined, but the translation has one undetermined dimension. For two such pairs of rigid transformations A_1, B_1 and A_2, B_2, we can construct three orthogonal directions using the (mutual) screw axis direction, the direction separating the two screw axes, and their cross product. We can then solve for the rotation as before.

Since R_A always has at least one eigenvalue of unity, $(R_A - I)$ in the translation equation 18 has a one-dimensional null space, and the solution space has one undetermined dimension. Namely, this is the common screw axis direction. Since all the screws are parallel, this null space is the same for all of the equations. In this case, we attempt to keep the translation as close to the origin as possible, which we do by adding another row to the matrix $\mathbf{r}_a^T \mathbf{t}_x = 0$. Finally, we stack the equations produced by each pair of $AX = XB$ equations to produce a single least-squares solution for $\mathbf{t}_x$.

Note that this gives an alternative method for solving the skew screw axis case: choose one screw axis as one direction, the direction along the shortest line between two screw axes as another, and their cross product as the third. Since in this case the null space of $(R_A - I)$ for different equations are not the same, the translation has a unique solution.

Coincident screw axes (2 undetermined dimensions). In the coincident screw axis case, only one direction for the As and for the Bs can be determined. The requirement that R_X rotate one direction onto another lets us solve for two rotational dimensions, but the third dimension is undetermined. In this case we choose the rotation axis to be along the (average) cross product of the screw axes of each A_i, B_i pair, which minimizes the rotation angle. We then solve for the translation as per the parallel axis case.

A.5 Cases without rotation

In some cases we may not have rotations at all; here the screw axis has direction equal to the translation but its offset perpendicular to the axis direction is undefined. As such, we may be able to construct axis directions, but the origin position of the axes are undetermined. In terms of the separated equations, the rotation equation 17 reduces to $R_X = R_X$, which gives us no information. The left side of the translation equation 18 is zero, which means that $\mathbf{t}_x$ cannot be determined at all, but there may be some information about R_X.

Nonparallel translation (3 undetermined dimensions). If the translations are nonparallel, we can use two non-parallel translations and their cross product to construct three mutually orthogonal axis directions. This allows us to determine the rotation uniquely, as in the noncoincident screw axis cases. As noted above, the translation is completely undetermined.

Parallel translation case (4 undetermined dimensions). Similarly, the parallel translation case is analogous to the coincident screw axis case in that the rotation can be determined up to one dimension. As in the coincident screw axis case, the requirement that R_X rotate one direction onto another lets us solve for two rotational dimensions, but the third dimension is undetermined. We can use the same method as the coincident screw axis case to determine a feasible rotation. As in the nonparallel translation case, the translation is completely undetermined.

No information case (6 undetermined dimensions). If we have no information at all (no equations, or all As and Bs are the identity), then we simply guess X to be the identity.

A.6 Omitted cases

(Chen, 1991) addresses the cases where some or all of the rotations are 180 degrees. We have not implemented these cases so far because they are rare in practical use. Typically, the As and Bs in the equations result from the relationship between two positions of a joint; this means skew screw axes for a ball joint, coincident screw axes for a revolute or cylindrical joint, and parallel translation for a prismatic joint. Therefore, it is rare to find cases with only 180-degree rotations or mixed types of equations.

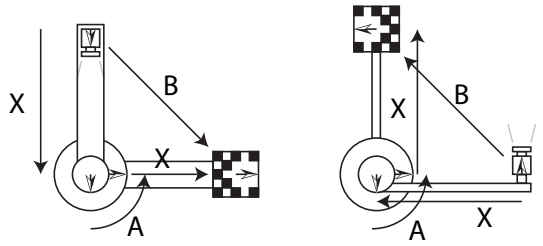


Fig. 10 Two-fold ambiguity for $XAX = B$ illustrated using a camera, a calibration target, and a revolute joint. A is a pose measurement taken by a revolute joint, B is a pose measurement taken by a camera and a checkerboard, and the two Xs are known to be equal. There are two distinct possible values of X that are consistent with the measurements. On the left, X has no rotational component, while on the right, X has a 180-degree rotation.

B Complexity

In this section we show that SL-R is NP-hard. Our proof is of weak NP-hardness due to the assumption that a transformation matrix may be specified in constant space. Furthermore, we show that the uniqueness variant of SL-R—that is, given an instance of SL-R for which a solution is guaranteed to exist, does there exist more than one solution?— is also NP-hard. Therefore, if P $\neq$ NP, there exists no polynomial-time algorithm that can solve SL-R, nor determine whether an instance of SL-R is ambiguous.

Our proof has two parts. First, we construct a gadget that forces any solution to make a binary choice. Then, we use this gadget to construct a reduction from the NP-complete (unique) partition problem.

B.1 Binary choice via XAX = B

We begin by inventing a SL-R gadget that forces the solution to make a binary choice. This gadget relies on equations of the form $XAX = B$, where all three quantities are rigid transformations and X is to be solved for.

In terms of SL-R such an equation consists of four absolute poses i, j, k, ℓ with relative pose constraints P_{ik} and $P_{j\ell}$, and a rigidity constraint X on the relative poses P_{ji} and $P_{k\ell}$.

Pictorially, such an equation looks like this:

$$\begin{array}{ccc} i & \xleftarrow{X=P_{ji}} & j \\ {\scriptstyle A=P_{ik}}\downarrow & & \downarrow{\scriptstyle B=P_{j\ell}} \\ k & \xrightarrow[X=P_{k\ell}]{} & \ell \end{array} \tag{21}$$

Mathematical properties. Inspiration taken from (Heath, 2008).

Suppose we represent each relative pose as a 4x4 rigid transformation matrix

$$\begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{22}$$

where R is a 3x3 rotation matrix, and $\mathbf{t}$ is a 3-vector representing translation.

From a single equation we have

$$XAX = B \tag{23}$$

$$AXAX = AB \tag{24}$$

$$AX = (AB)^{\frac{1}{2}} \tag{25}$$

Suppose we consider only the rotational part. For general real matrices, the square root has many possible values; however, since our solution is restricted to rotation matrices, this limits the possibilities. If $R_A R_B = I$, then I and any 180-degree rotation is a possible square root. Otherwise, $R_A R_B$ has a defined axis of rotation, and $(R_A R_B)^{\frac{1}{2}}$ must share that same axis, since applying any rotation twice preserves the axis of rotation. The magnitude of the rotation must be half that of $R_A R_B$, plus an integer multiple of π. Since a difference of 2π leaves the rotation matrix unchanged, this produces two possible values for $(R_A R_B)^{\frac{1}{2}}$ and therefore two possible values for R_X.

From here, we examine the translational part given the rotation. Denote the rotational and translational parts by R_X and $\mathbf{t}_x$ respectively. Then solving for the unknown $\mathbf{t}_x$ reduces to

$$R_X R_A \mathbf{t}_x + R_X \mathbf{t}_a + \mathbf{t}_x = \mathbf{t}_b \tag{26}$$

$$(R_X R_A + I)\, \mathbf{t}_x = \mathbf{t}_b - R_X \mathbf{t}_a \tag{27}$$

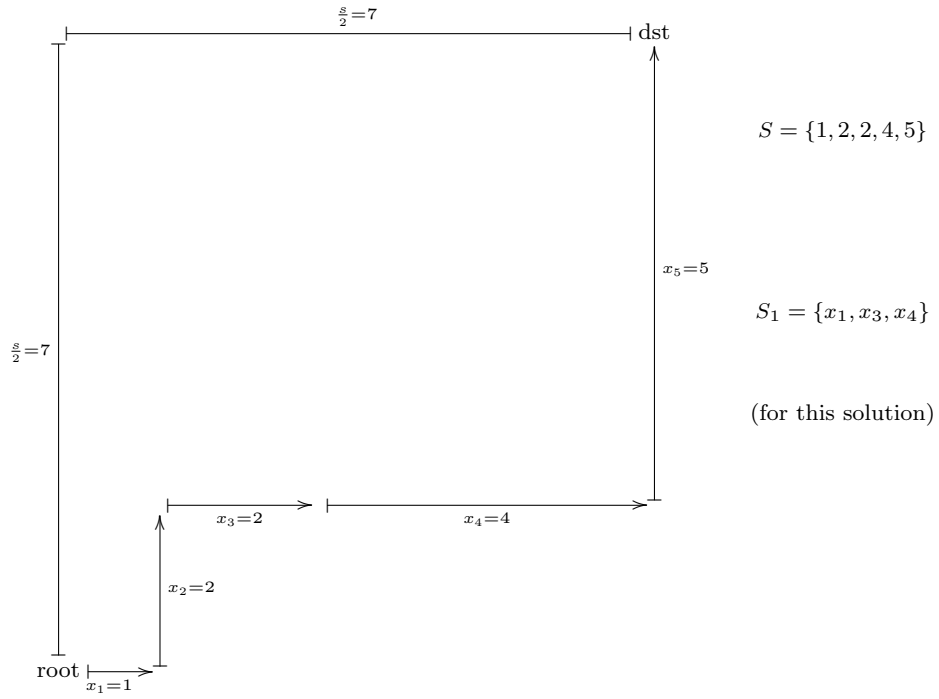


Fig. 11 Pictorial example of a PP reduction to SL-R and its solution. We encode PP as a Manhattan walk where steps to the east represent one partition and steps to the north the other partition. We enforce the choice between these two directions using gadgets of the form $XAX = B$.

This has a unique solution as long as $R_X R_A + I$ is invertible. To see when this is the case, consider this product for a unit vector x:

$$x^{-1} (R_X R_A + I) x = x^{-1} R_X R_A x + 1 \tag{28}$$

The right side is only nonpositive when $R_X R_A$ is a 180-degree rotation. Apart from this case, $R_X R_A + I$ is positive-definite and therefore invertible. Thus, unless $R_X R_A$ is a 180-degree rotation, $\mathbf{t}_x$ has a unique solution given R_X.

The gadget. Suppose we have a cycle of rigidity constraints X and relative pose constraints A and B that produces an $XAX = B$ equation. Let the rotational parts of A and B equal

$$R_A = R_B = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{29}$$

From this we have

$$R_X = R_A^{-1} (R_A R_B)^{\frac{1}{2}} \in \{I, I'\} \tag{30}$$

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{31}$$

$$I' = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{32}$$

Now let us examine the translational parts of A, B, and X. Suppose

$$\mathbf{t}_a = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{33}$$

$$\mathbf{t}_b = \begin{bmatrix} b \\ b \\ 0 \end{bmatrix} \tag{34}$$

for some constant b. In the case where $R_X = I$, we have

$$(R_X R_A + I) \mathbf{t}_x = \mathbf{t}_b - R_X \mathbf{t}_a \tag{35}$$

$$(R_A + I) \mathbf{t}_x = \mathbf{t}_b \tag{36}$$

$$\begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \mathbf{t}_x = \begin{bmatrix} b \\ b \\ 0 \end{bmatrix} \tag{37}$$

$$\mathbf{t}_x = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \tag{38}$$

In the case where $R_X = I'$, we have

$$(R_X R_A + I) \mathbf{t}_x = \mathbf{t}_b \tag{39}$$

$$\begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \mathbf{t}_x = \begin{bmatrix} b \\ b \\ 0 \end{bmatrix} \tag{40}$$

$$\mathbf{t}_x = \begin{bmatrix} 0 \\ b \\ 0 \end{bmatrix} \tag{41}$$

Therefore, in our gadget, X either represents a step "forwards" ($+x$ direction) of size b, or a step to the "left" ($+y$ direction) of size b followed by an about-face. If $b = 0$, then the choice is between standing still and an about-face. A pictorial representation is shown in Figure 10. Also note that if b is an integer, all arithmetic here is integer, which shows that the NP-hardness of SL-R is not (solely) due to real arithmetic.

B.2 Reduction from the (unique) partition problem

With this gadget in hand, we now show a polynomial reduction from the (unique) partition problem to the (unique) sensor localization with rigidity problem.

The partition problem. Recall the partition problem (PP). The definition of the problem is as follows: given a set of positive integers $S = \{x_1, x_2, \dots x_n\}$, determine if there exists a subset of the integers S_1 such that

$$\sum_{x \in S_1} x = \sum_{x \notin S_1} x \tag{42}$$

The standard form of the problem is well-known to be NP-complete; (Dieudonné et al, 2010) shows that, given an instance of this problem for which it is guaranteed that at least one solution exists, the problem of determining whether there exists more than one solution (counting $S - S_1$ to be the same solution as S_1) is also NP-complete. They call this version the unique partition problem (UPP).

We likewise define the uniqueness version (USL-R) of SL-R to be: given an instance for SL-R which a solution is guaranteed to exist, does there exist more than one solution?

Integer representation. We represent the integers of PP in SL-R as follows. For each $x_i \in S$, we introduce two rigidity constraints X_i and X_i'. For both of these we construct an $XAX = B$ cycle of the form described above. For the one with X_i, we set $b = x_i$; for the one with X_i', we set $b = 0$. This produces four options for the product $X_i X_i'$, corresponding to our choice of picking $R_X = I$ or $R_X = I'$ for each rotational part:

1. A step forward of size x_i (choose $R_{X_i} = I$ and $R_{X_i'} = I$).
2. A step forward of size x_i followed by an about-face (choose $R_{X_i} = I$ and $R_{X_i'} = I'$).
3. A step left of size x_i (choose $R_{X_i} = I'$ and $R_{X_i'} = I'$).
4. A step left of size x_i followed by an about-face (choose $R_{X_i} = I'$ and $R_{X_i'} = I$).

We construct each such cycle so that they share a common "root" absolute pose, so these are the only choices to be made here. Since we do constant work and produce a constant number of poses, relative pose constraints, and rigidity constraints per element of S, this part of the reduction is linear in time and problem size.

A Manhattan walk. With our integer representation in place, we introduce one more cycle, which we will treat as two paths from the root to a destination. On one side we constrain the absolute pose of the the destination to equal

$$P_{\text{dst}} = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{2}s \\ 0 & 1 & 0 & \frac{1}{2}s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{43}$$

$$\text{where } s = \sum_i^n x_i \tag{44}$$

On the other side we have a sequence of $2n$ poses with relative poses labeled such that they form the product

$$P_{\text{dst}} = \prod_i^n X_i X_i' = X_1 X_1' X_2 X_2' \dots X_n X_n' \tag{45}$$

This part of the problem reduction is also linear in time and problem size.

The intuition is that we have constructed a problem where the goal is to reach a destination $\frac{s}{2}$ north and $\frac{s}{2}$ east of the starting point, using a set of steps of size $x_1, x_2, \dots x_n$.

At each two-pose step towards the destination $X_i X_i'$ a solution must take one of the four options enumerated earlier. This constrains the translations to lie only on cardinal directions. Since the Manhattan distance to our target is s and there is only have a total of s distance in translations in the product of rigid transformations, any solution must step towards the destination in every step. The solution cannot do an about-face before the last step, since it would have to step away from the destination on the next step, and the solution cannot do an about-face on the last step, since it would be facing the wrong direction at the end. Therefore any solution must consist of only options 1 and 3, with forward steps being to the "east" and left steps to the "north". Since the east steps total the same distance as the north steps in any solution, this forms a one-to-one correspondence to a solution to PP—the indices of the steps in which the solution moved east correspond to the indices of the integers in one half of the partition. To avoid the duplicate solution where S_1 is swapped with $S - S_1$, we require the first step to be east by adding a relative pose constraint.

An example is shown in Figure 11.

Therefore PP reduces to SL-R, UPP reduces to USL-R, and both SL-R and USL-R are NP-hard.