Optimal and Robust Category-level Perception: Object Pose and Shape Estimation from 2D and 3D Semantic Keypoints

Jingnan Shi, Heng Yang, Luca Carlone

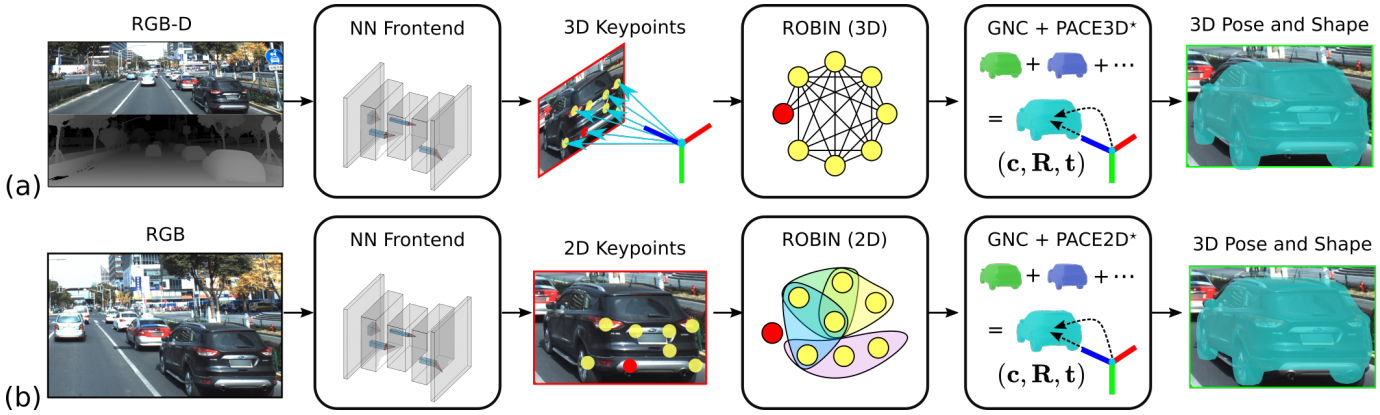


Fig. 1. We develop algorithms for 3D-3D and 2D-3D category-level perception, which estimate 3D pose and shape of an object from 3D and 2D sensor data, respectively. (a) PACE3D# performs 3D-3D category-level perception and works on RGB-D inputs. We pass the sensor data through a neural network front-end to obtain 3D keypoint detections. We then use ROBIN with 3D-3D compatibility (hyper)graphs (Section V) to prune outliers (outliers are shown as red dots, while inliers are yellow). We finally pass the resulting measurements to our optimal solver (PACE3D*, Section VI-A) wrapped in a graduated non-convexity (GNC) scheme (Section VII), which estimates the object pose and shape coefficients. (b) PACE2D# performs 2D-3D category-level perception and works on RGB inputs. The images are passed through a neural network to obtain 2D keypoints. We then use ROBIN with 2D-3D compatibility hypergraphs (Section V) to prune outliers. We finally pass the resulting measurements to our optimal solver (PACE2D*, Section VI-B) with GNC to obtain a pose and shape estimate.

***Abstract*—We consider a *category-level perception* problem, where one is given 2D or 3D sensor data picturing an object of a given category (*e.g.,* a car), and has to reconstruct the 3D pose and shape of the object despite intra-class variability (*i.e.,* different car models have different shapes). We consider an *active shape model*, where —for an object category— we are given a library of potential CAD models describing objects in that category, and we adopt a standard formulation where pose and shape are estimated from 2D or 3D keypoints via non-convex optimization. Our first contribution is to develop PACE3D* and PACE2D*, the first *certifiably optimal* solvers for pose and shape estimation using 3D and 2D keypoints, respectively. Both solvers rely on the design of tight (*i.e.,* exact) semidefinite relaxations. Our second contribution is to develop outlier-robust versions of both solvers, named PACE3D# and PACE2D#. Towards this goal, we propose ROBIN, a general graph-theoretic framework to prune outliers, which uses *compatibility hypergraphs* to model measurements' compatibility. We show that in category-level perception problems these hypergraphs can be built from winding orders of the keypoints (in 2D) or their convex hulls (in 3D), and many outliers can be pruned via maximum hyperclique computation. The last contribution is an extensive experimental evaluation. Besides providing an ablation study on simulated datasets and on the PASCAL3D+ dataset, we combine our solver with a deep keypoint detector, and show that PACE3D# improves over the state of the art in vehicle pose estimation in the ApolloScape datasets, and its runtime is compatible with practical applications.**

***Index Terms*—category-level perception, outliers-robust estimation, certifiable algorithms.**

I. Introduction

Robotics applications, from self-driving cars to domestic robotics, demand robots to be able to identify and estimate the pose and shape of nearby objects. In self-driving applications, the perception system needs to estimate the poses of other vehicles, identify traffic lights and signs, and detect pedestrians. Similarly, domestic applications require estimating the location and shape of objects to support effective interaction and manipulation [60], [32], [73]. Object pose estimation is made harder by the large intra-class shape variability of common objects: for instance, the shape of a car largely varies depending on the model (*e.g.,* take a van versus a sedan).

Despite the fast-paced progress, reliable 3D object pose estimation remains a challenge, as witnessed by recent self-driving car accidents caused by misdetections [63]. Deep learning has been making great strides in enabling robots to detect objects; popular tools such as YOLO [78] and Mask-RCNN [36] have made object detection possible on commodity hardware and with reasonable performance for in-distribution test data. However, detections are typically at the level of categories (*e.g.,* car) rather than at the level of instances (*e.g.,* a specific car model). In turn, category-level perception renders the use of standard tools for pose estimation (from point cloud registration [121], [37], [70] to the Perspective-n-Point problem [123], [42], [87]) ineffective, since they rely on the knowledge of the shape of the object.

These limitations have triggered robotics and computer vision research on category-level 3D object pose and shape estimation (see Section IX for an in-depth review). Traditional methods include the popular *active shape model* [19], [125], [117], where one attempts to estimate the pose and shape of an object given a large database of 3D CAD models. Despite its popularity (*e.g.,* the model is also used in human shape estimation and face detection [125]), estimation with active shape models leads to a non-convex optimization problem and local solvers get stuck in poor solutions, and are sensitive to outliers [125], [117]. More recently, research effort has

J. Shi, H. Yang, and L. Carlone are with the Laboratory for Information & Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139, USA, Email: {jnshi,hankyang,lcarlone}@mit.edu

been devoted to end-to-end learning-based 3D pose estimation with encouraging results in human pose estimation [44] and vehicle pose estimation [15], [41], [57], [46], [97]; these approaches still require a large amount of 3D labeled data, which is (or expensive) to obtain in the wild.

Contribution. We address the shortcomings of existing approaches for pose and shape estimation based on the active shape model and propose the first approaches that can compute optimal estimates and are robust to a large number of outliers. We consider a category-level perception problem, where one is given keypoint detections of an object belonging to a given category (*e.g.,* detections of the wheels, rear-view mirrors, and other interest points of a car), and has to reconstruct the pose and shape of the object. We assume the availability of a library of CAD models of objects in that category; such a library is typically available, since CAD models are extensively used in the design, manufacturing, and simulation of 3D objects.

Our first contribution is to develop the first *certifiably optimal* solvers for pose and shape estimation using 3D and 2D keypoints. In the 3D case, we show that —despite the non-convexity of the problem— rotation estimation can be decoupled from the estimation of object translation and shape, and we demonstrate that (i) the optimal object rotation can be computed via a tight (small-size) semidefinite relaxation, and (ii) the translation and shape parameters can be computed in closed form given the rotation. We call the resulting solver PACE3D* (*Pose and shApe estimation for 3D-3D Category-level pErception*). In the 2D case, we formulate pose and shape estimation using an algebraic point-to-line cost, and leverage *Lasserre's hierarchy of semidefinite relaxations* [49] to solve the problem to certifiable global optimality. We call the resulting solver PACE2D* (*Pose and shApe estimation for 2D-3D Category-level pErception*). Contrarily to PACE3D*, PACE2D* leads to semidefinite relaxations whose size increases with the number of CAD models in the active shape model.

Our second contribution is to develop an outlier rejection scheme applicable to both PACE3D* and PACE2D*. Towards this goal, we introduce a general framework for graph-theoretic outlier pruning, named ROBIN, which generalizes our previous work [121], [90] to use hypergraphs. ROBIN models compatibility between subset of measurements using a *compatibility hypergraph*. We show that the compatibility hypergraph can be efficiently constructed by inspecting the winding orders of the keypoints in 2D, or the convex hulls of the keypoints in 3D. We then prove that all the inliers are contained in a single hyperclique of the compatibility hypergraph and can be typically found within the maximum hyperclique. ROBIN is able to remove a large fraction of outliers. The resulting measurements are then passed to our optimal solvers (PACE3D* and PACE2D*), that we also wrap in a standard *graduated non-convexity* [115] scheme to mitigate the impact of outliers that survived ROBIN. The resulting outlier-robust approaches are named PACE3D# and PACE2D# and are illustrated in Fig. 1.

Our last contribution is an extensive experimental evaluation in both synthetic experiments and real datasets [110]. We provide an ablation study on simulated datasets and on the PASCAL3D+ dataset, and show that (i) PACE3D* is more accurate than state-of-the-art iterative solvers, (ii) PACE2D* is more accurate than baseline local solvers and convex relaxations based on the weak perspective projection model [125], [117], (iii) PACE3D# dominates other robust solvers and is robust to $70-90\%$ outliers, and (iv) PACE2D# is robust to 20% outliers. Finally, we integrate our solvers in a realistic system —including a deep keypoint detector— and apply it to vehicle pose and shape estimation in the ApolloScape [110] driving datasets. While PACE2D# is currently slow and suffers from the low quality of the deep keypoint detections, PACE3D# largely outperforms the state of the art and a non-optimized implementation runs in a fraction of a second. We also show that ROBIN is even able to detect mislabeled keypoints used to train the keypoint detector in the ApolloScape dataset [110].

Novelty with Respect to [90], [89]. In our previous works, we introduced ROBIN [90], a graph theoretic outlier rejection framework, and two solvers [89] (PACE3D* and PACE3D#) for pose and shape estimation from 3D keypoints. The present paper (i) generalizing ROBIN to handle combinatorial *compatibility tests*, and (ii) extends the concept of inlier selection to maximum hypercliques on *compatibility hypergraphs*. This paper also develops PACE2D* and PACE2D#, which estimate pose and shape from only 2D (instead of 3D) keypoints. In addition, we report a more comprehensive experimental evaluation in simulation and on the ApolloScape dataset.

Paper Structure. Section II formulates the category-level perception problem. Section III provides a brief overview of the proposed approaches (also summarized in Fig. 1). Sections IV and V present our graph-theoretic outlier pruning (ROBIN) and its application to category-level perception. Section VI introduces our certifiably optimal solvers for category-level perception and Section VII recalls how to wrap the solvers in a graduated non-convexity scheme. Section VIII discusses experimental results. Section IX provides an in-depth review of related work. Section X concludes the paper.

II. Problem Statement: 3D-3D and 2D-3D Category-Level Perception

In this section, we formulate the 3D-3D and 2D-3D category-level perception problems. The goal is to compute the 3D pose and shape of an object, given 3D or 2D sensor data. We focus on a *multi-stage* setup where a front-end is used to extract 2D or 3D semantic keypoints from the sensor data, which are then used by a back-end solver to estimate the object's 3D pose $(\boldsymbol{R}, \boldsymbol{t})$ and shape, where $\boldsymbol{R} \in \mathrm{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$ are 3D rotation and translation, respectively. The front-end is typically implemented using standard deep networks [73], [86], while our goal here is to design more accurate and robust back-ends.

In the following, we first introduce a standard parametrization of the object shape (Section II-A), and then formalize the 3D-3D category-level perception problem (Section II-B) and its 2D-3D counterpart (Section II-C).

A. Active Shape Model

We assume the object shape to be partially specified: we are given a library of 3D CAD models $\mathcal{B}^k$, $k = 1, \ldots, K$, and assume that the unknown object shape $\mathcal{S}$ (modeled as a collection of 3D points) can be written as a combination of the points on the given CAD models. More formally, each point $\boldsymbol{s}_i$ of the shape $\mathcal{S}$ can be written as:

$$\boldsymbol{s}_i = \textstyle\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k \tag{1}$$

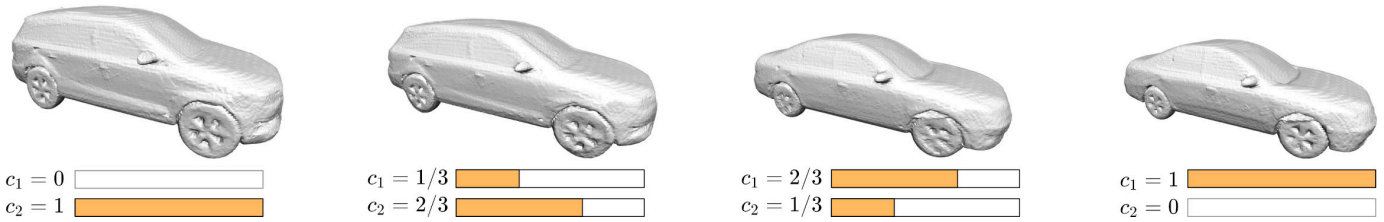


Fig. 3. Illustration of a simple active shape model with two CAD models: a Hyundai Sonata (with shape coefficient c_1) and an Audi Q7 (with shape coefficient c_2). From left to right, we show four convex combinations of the CAD models, with c_1 increasing and c_2 decreasing while keeping $c_1 + c_2 = 1$. The shape variations between the four combinations show the expressivity of the active shape model and its capability to interpolate between shapes.

where $\boldsymbol{b}_i^k$ is a given point on the surface of the CAD model $\mathcal{B}^k$; the *shape parameters* $\boldsymbol{c} \triangleq [c_1 \dots c_K]^\mathsf{T}$ are unknown, and the entries of $\boldsymbol{c}$ are assumed to be non-negative and sum up to 1, *i.e.,* $\boldsymbol{c}$ belongs to the K-dimensional probability simplex $\Delta_K := \{\boldsymbol{c} \in \mathbb{R}^K \mid \boldsymbol{c} \geq \boldsymbol{0}, \sum_{k=1}^K c_k = 1\}$. For instance, if — upon estimation— the vector $\boldsymbol{c}$ has the l-th entry equal to 1 and the remaining entries equal to zero in (1), then the estimated shape matches the l-th CAD model in the library; therefore, the estimation of the shape parameters $\boldsymbol{c}$ can be understood as a fine-grained classification of the object among the instances in the library. However, the model is even more expressive, since it allows the object shape to be a convex combination of CAD models, which enables the active shape model (1) to interpolate between different shapes in the library; see Fig. 3.

B. 3D-3D Category-Level Perception

In the 3D-3D category-level perception problem, the goal is to estimate an object's pose and shape, given a set of N 3D keypoint detections typically obtained using learning-based keypoint detectors. Such detectors are trained to detect semantic features of the 3D object (*e.g.,* wheels of a car), and can be applied to RGB-D or RGB+Lidar data (*e.g.,* [73]).

We assume each **3D measurement** $\boldsymbol{p}_{3D,i}$ $(i = 1, \dots, N)$ is a noisy measurement of a keypoint $\boldsymbol{s}_i$ of our target object, in the coordinate frame of the sensor. More formally, each $\boldsymbol{p}_{3D,i}$ is described by the following generative model:

$$\boldsymbol{p}_{3D,i} = \boldsymbol{R} \sum_{k=1}^K c_k \boldsymbol{b}_i^k + \boldsymbol{t} + \boldsymbol{\epsilon}_{3D,i} \qquad i = 1, \dots, N \quad (2)$$

where the measurement $\boldsymbol{p}_{3D,i}$ pictures a 3D point on the object (written as a linear combination $\sum_{k=1}^K c_k \boldsymbol{b}_i^k$ of the shapes in the library as in (1)), after the point is rotated and translated according to the 3D pose $(\boldsymbol{R}, \boldsymbol{t})$ of the object, and where $\boldsymbol{\epsilon}_{3D,i}$ represents measurement noise. Intuitively, each measurement corresponds to a noisy measurement of a semantic feature of the object (*e.g.,* wheel center or rear-view mirrors of a car) and each $\boldsymbol{b}_i^k$ corresponds to the corresponding feature (*e.g.,* wheel or mirror) location for a specific CAD model.

Problem 1 (Robust 3D-3D Category-Level Perception). *Compute the 3D pose* $(\boldsymbol{R}, \boldsymbol{t})$ *and shape* $(\boldsymbol{c})$ *of an object given* N *3D keypoint measurements in the form* (2)*, possibly corrupted by outliers,* i.e., *measurements with large error* $\boldsymbol{\epsilon}_{3D,i}$.

C. 2D-3D Category-Level Perception

In the 2D-3D category-level perception problem, we want to estimate an object's 3D pose and shape, given only 2D projections of keypoints. In this case, we describe each **2D measurement** using the following generative model:

$$\boldsymbol{p}_{2D,i} = \pi\left(\boldsymbol{R} \sum_{k=1}^K c_k \boldsymbol{b}_i^k + \boldsymbol{t}\right) + \boldsymbol{\epsilon}_{2D,i} \qquad i = 1, \dots, N \quad (3)$$

where $\boldsymbol{p}_{2D,i}$ represents a 2D (pixel) measurement, $\pi(\cdot)$ is the canonical perspective projection,[1] and $\boldsymbol{\epsilon}_{2D,i}$ is the measurement noise. Intuitively, the measurements in (3) correspond to pixel projections of the object keypoints onto an image.

Problem 2 (Robust 2D-3D Category-Level Perception). *Compute the 3D pose* $(\boldsymbol{R}, \boldsymbol{t})$ *and shape* $(\boldsymbol{c})$ *of an object given* N *2D keypoint measurements in the form* (3)*, possibly corrupted by outliers,* i.e., *measurements with large error* $\boldsymbol{\epsilon}_{2D,i}$.

III. Overview of PACE#: Pose and Shape Estimation for Robust Category-level Perception

Our approach, named PACE#, is summarized in Fig. 1, for both the 3D-3D case (PACE3D#, Fig. 1a) and the 2D-3D case (PACE2D#, Fig. 1b). We assume access to a perception front-end that detects semantic keypoints given sensor data. Our work forms the back-end, and consists of two stages. In the first stage, we employ a graph-theoretic framework, named ROBIN, to pre-process the keypoints and prune gross outliers without explicitly solving the underlying estimation problem. We then pass the filtered measurements to the second stage, where an optimal solver (wrapped in a graduated non-convexity scheme) computes a pose and shape estimate.[2]

In the following we introduce Stage 1 by first presenting ROBIN, a general framework for outlier pruning (Section IV) and then discussing its application to category-level perception (Section V); we then discuss Stage 2 by presenting our optimal solvers for 3D-3D (PACE3D⋆, Section VI-A) and 2D-3D (PACE2D⋆, Section VI-B) category-level perception, and a brief review of graduated non-convexity [115] (Section VII).

IV. Stage 1: Graph-theoretic Outlier Pruning with ROBIN

This section develops a general framework to prune gross outliers from a set of measurements without explicitly computing an estimate for the variables of interest. In particular, we introduce the notion of n-*invariants* to check if a subset of measurements contains outliers. We then use these checks to construct *compatibility hypergraphs* that describe mutually compatible measurements, and show how to reject outliers by

[1] For a 3D vector $\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$, the canonical projection is $\pi(\boldsymbol{p}) = \begin{bmatrix} p_x/p_z \\ p_y/p_z \end{bmatrix}$.

[2] We use a two-stage approach (*i.e.,*ROBIN followed by graduated non-convexity) for two reasons. While our solver with graduated non-convexity is robust against 10% of outliers in 2D-3D problems and $50-60\%$ of outliers in 3D-3D problems, our goal is to further increase its robustness. Indeed we show that ROBIN boosts robustness to 20% in 2D-3D problems and $70-90\%$ in 3D-3D problems. In addition, since the first stage prunes outliers independently from the solver, ROBIN can be used in a plug-and-play manner with other existing solvers to boost their robustness.

computing maximum hypercliques of these graphs. Combining these insights, we obtain ROBIN (*Reject Outliers Based on INvariants*), our graph-theoretic algorithm for pruning outliers.

This section presents our framework in full generality and then we tailor it to category-level perception in Section V. In particular, here we consider a more general measurement model that relates measurements $\boldsymbol{y}_i$ to the to-be-estimated variable $\boldsymbol{x} \in \mathbb{X}$ (where $\mathbb{X}$ is the domain of $\boldsymbol{x}$, *e.g.,* the set of 3D poses) and a given model $\boldsymbol{\theta}_i$ (*e.g.,* our CAD models):

$$\boldsymbol{y}_i = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\theta}_i, \boldsymbol{\epsilon}_i), \qquad i = 1, \dots, N \tag{4}$$

where $\boldsymbol{\epsilon}_i$ denotes the measurement noise. Clearly, eqs. (2) and (3) can be understood as special instances of (4), where $\boldsymbol{x}$ includes the unknown pose and shape of the object, and the (given) model $\boldsymbol{\theta}$ corresponds to the CAD models.

A. From Measurements to Invariants

This section formalizes the concepts of n-invariant and generalized n-invariant, which are the building blocks of our outlier pruning framework. The main motivation is to use invariance to establish checks on the (inlier) measurements that hold true regardless of the state under estimation; we are later going to use these checks to detect outliers.

Let us consider the measurements in eq. (4) and denote the indices of the measurements as $\mathcal{Y} \doteq \{1, \dots, N\}$. For a given integer $n \leq N$, let $\mathcal{M} \subset \mathcal{Y}$ be a subset of n indices in $\mathcal{Y}$, and denote with $\mathcal{M}_j$ the j-th element of this subset (with $j = 1, \dots, n$). Then, we use the following notation:

$$\boldsymbol{y}_{\mathcal{M}} = \begin{bmatrix} \boldsymbol{y}_{\mathcal{M}_1} \\ \boldsymbol{y}_{\mathcal{M}_2} \\ \vdots \\ \boldsymbol{y}_{\mathcal{M}_n} \end{bmatrix}, \quad \boldsymbol{\theta}_{\mathcal{M}} = \begin{bmatrix} \boldsymbol{\theta}_{\mathcal{M}_1} \\ \boldsymbol{\theta}_{\mathcal{M}_2} \\ \vdots \\ \boldsymbol{\theta}_{\mathcal{M}_n} \end{bmatrix}, \quad \boldsymbol{\epsilon}_{\mathcal{M}} = \begin{bmatrix} \boldsymbol{\epsilon}_{\mathcal{M}_1} \\ \boldsymbol{\epsilon}_{\mathcal{M}_2} \\ \vdots \\ \boldsymbol{\epsilon}_{\mathcal{M}_n} \end{bmatrix} \tag{5}$$

which is simply stacking together measurements $\boldsymbol{y}_i$, parameters $\boldsymbol{\theta}_i$, and noise $\boldsymbol{\epsilon}_i$ for the subset of measurements $i \in \mathcal{M}$.

Let us now formalize the notion of noiseless invariance.

Definition 1 (Noiseless n-Invariant). *Consider the generative model (4) and assume there is no measurement noise (*i.e., *$\boldsymbol{y}_i = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\theta}_i)$). Then a function $\boldsymbol{f}$ is called a noiseless* n-invariant *if for any arbitrary $\mathcal{M} \subset \mathcal{Y}$ of size n, the following relation holds, regardless of the choice of $\boldsymbol{x}$:*

$$\boldsymbol{f}(\boldsymbol{y}_{\mathcal{M}}) = \boldsymbol{f}(\boldsymbol{\theta}_{\mathcal{M}}) \tag{6}$$

Intuitively, an n-invariant function $\boldsymbol{f}$ takes a subset of models $\boldsymbol{\theta}_{\mathcal{M}}$ and computes a quantity $\boldsymbol{f}(\boldsymbol{\theta}_{\mathcal{M}})$ that remains constant when the models are transformed by the measurement model $\boldsymbol{h}$ to generate the measurements $\boldsymbol{y}_{\mathcal{M}}$.

Example. To concretely understand invariance, consider a simpler instance of (2) where there is a single shape:

$$\boldsymbol{y}_i = \boldsymbol{R}\boldsymbol{\theta}_i + \boldsymbol{t} + \boldsymbol{\epsilon}_i \qquad i = 1, \dots, N \tag{7}$$

Eq. (7) is also known as the *point cloud registration* problem [2], [37] and consists in finding a rigid body transformation $(\boldsymbol{R}, \boldsymbol{t})$ (where $\boldsymbol{R} \in \mathrm{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$) that aligns two sets of 3D points $\boldsymbol{y}_i \in \mathbb{R}^3$ and $\boldsymbol{\theta}_i \in \mathbb{R}^3$, with $i = 1, \dots, N$. The corresponding measurement model can again be seen to be an instance of the general model (4).

In the noiseless case ($\boldsymbol{\epsilon}_i = \boldsymbol{0}$), it follows from (7) that

$$\begin{aligned} \|\boldsymbol{y}_j - \boldsymbol{y}_i\| &= \|(\boldsymbol{R}\boldsymbol{\theta}_j + \boldsymbol{t}) - (\boldsymbol{R}\boldsymbol{\theta}_i + \boldsymbol{t})\| = \\ &= \|\boldsymbol{R}(\boldsymbol{\theta}_j - \boldsymbol{\theta}_i)\| = \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| \end{aligned} \tag{8}$$

for any pair of measurements i, j, where $\|\cdot\|$ is the 2-norm and we used the fact that the 2-norm is invariant under rotation. This is an example of noiseless 2-invariant, $\boldsymbol{f}(\boldsymbol{y}_i, \boldsymbol{y}_j) \doteq \|\boldsymbol{y}_j - \boldsymbol{y}_i\| = \boldsymbol{f}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, which relates measurements and model regardless of the choice of $\boldsymbol{x}$, and formalizes the intuition that the distance between pairs of points in a point cloud is invariant under rigid transformations.

While Definition 1 provides a general definition of noiseless invariance, practical problems always involve noise. Therefore we need to generalize the notion of invariance as follows.

Definition 2 (Generalized n-Invariant). *Given eq. (4) and assuming the measurement noise is bounded $\|\boldsymbol{\epsilon}_i\| \leq \beta$ ($i = 1, \dots, N$), a pair of functions $(\boldsymbol{f}, \boldsymbol{F})$ is called a* generalized n-invariant *if for any arbitrary $\mathcal{M} \subset \mathcal{Y}$ of size n, the following relation holds, regardless of the choice of $\boldsymbol{x}$:*

$$\boldsymbol{f}(\boldsymbol{y}_{\mathcal{M}}) \in \boldsymbol{F}(\boldsymbol{\theta}_{\mathcal{M}}, \beta) \tag{9}$$

where $\boldsymbol{F}(\boldsymbol{\theta}_{\mathcal{M}}, \beta)$ is a set-valued function (independent on $\boldsymbol{x}$).

Intuitively, because of the noise, now the measurements can produce a number of different invariants $\boldsymbol{f}(\boldsymbol{y}_{\mathcal{M}})$, but we can still define a set $\boldsymbol{F}(\boldsymbol{\theta}_{\mathcal{M}}, \beta)$ that contains all potential invariants produced by the measurements. An example is in order.

Example. Going back to the example of point cloud registration in eq. (7), with $\|\boldsymbol{\epsilon}_i\| \leq \beta$ and $\|\boldsymbol{\epsilon}_j\| \leq \beta$, we have

$$\|\boldsymbol{y}_j - \boldsymbol{y}_i\| = \|\boldsymbol{R}(\boldsymbol{\theta}_j - \boldsymbol{\theta}_i) + \boldsymbol{\epsilon}_j - \boldsymbol{\epsilon}_i\| \tag{10}$$

If we apply the triangle inequality to the right-hand-side of (13) we obtain:

$$\begin{aligned} \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| - \|\boldsymbol{\epsilon}_j - \boldsymbol{\epsilon}_i\| &\leq \overbrace{\|\boldsymbol{R}(\boldsymbol{\theta}_j - \boldsymbol{\theta}_i) + \boldsymbol{\epsilon}_j - \boldsymbol{\epsilon}_i)\|}^{= \|\boldsymbol{y}_j - \boldsymbol{y}_i\| \text{ per eq. (13)}} \\ &\leq \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| + \|\boldsymbol{\epsilon}_j - \boldsymbol{\epsilon}_i\| \end{aligned} \tag{11}$$

Now $\|\boldsymbol{\epsilon}_i\| \leq \beta$ and $\|\boldsymbol{\epsilon}_j\| \leq \beta$ imply $\|\boldsymbol{\epsilon}_j - \boldsymbol{\epsilon}_i\| \leq 2\beta$. Substituting in (11) we obtain:

$$\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| - 2\beta \leq \|\boldsymbol{y}_j - \boldsymbol{y}_i\| \leq \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| + 2\beta \tag{12}$$

or, in other words:

$$\overbrace{\|\boldsymbol{y}_j - \boldsymbol{y}_i\|}^{\boldsymbol{f}(\boldsymbol{y}_{\mathcal{M}})} \in \overbrace{\left[\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| - 2\beta, \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| + 2\beta\right]}^{\boldsymbol{F}(\boldsymbol{\theta}_{\mathcal{M}}, \beta)} \tag{13}$$

which corresponds to our definition of generalized n-invariant (with $n = 2$). Geometrically, eq. (13) states the distances between pairs of measured points ($\boldsymbol{y}_j$ and $\boldsymbol{y}_i$) must match corresponding distances between points in our model ($\boldsymbol{\theta}_j$ and $\boldsymbol{\theta}_i$) up to noise. Note that when $\beta = 0$ (noiseless case), eq. (13) reduces back to eq. (8), as the set $\boldsymbol{F}(\boldsymbol{\theta}_{\mathcal{M}}, \beta) \doteq [\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| - 2\beta, \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| + 2\beta]$ reduced to the singleton $\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\|$.

Definition 2 generalizes Definition 1 to account for the presence of noise. Moreover, as we will see in Section V, we can develop generalized n-invariants for category-level perception even when noiseless invariants are difficult to find. In other words, while it may be difficult to pinpoint a noiseless invariant function, it is often easier to find a set of values that a suitable function $\boldsymbol{f}(\boldsymbol{y}_{\mathcal{M}})$ must belong to. In the following sections unless otherwise specified, we use the term n-invariants to refer to generalized n-invariants.

B. From Invariants to Compatibility Tests for Outlier Pruning

While the previous section developed invariants without distinguishing inliers from outliers, this section shows that the resulting invariants can be directly used to check if a subset of measurements contains an outlier. Towards this goal, we formalize the notion of inlier and outlier.

Definition 3 (Inliers and Outliers). *Given measurements (4) and a threshold $\beta \geq 0$, a measurement i is an* inlier *if the corresponding noise satisfies $\|\boldsymbol{\epsilon}_i\| \leq \beta$ and is an* outlier *otherwise.*

The notion of invariants introduced in the previous section allowed us to obtain relations that depend on the measurements and a noise bound, but are independent on $\boldsymbol{x}$, see eq. (9). Therefore, we can directly use these relations to check if a subset of n measurements contains outliers: if eq. (9) is not satisfied by a subset of measurements $\boldsymbol{y}_{\mathcal{M}}$ then the corresponding subset of measurements *must* contain an outlier. We call the corresponding check a *compatibility test*. In the following, we provide an example of compatibility test for point cloud registration. The reader can find more examples of compatibility tests for other applications in [90].

Example of Compatibility Test. For our point cloud registration example, eq. (13) states that any pair of measurements $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ with noise $\|\boldsymbol{\epsilon}_i\| \leq \beta$ and $\|\boldsymbol{\epsilon}_j\| \leq \beta$ must satify:

$$\|\boldsymbol{y}_j - \boldsymbol{y}_i\| \in \left[\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| - 2\beta, \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i\| + 2\beta \right] \tag{14}$$

If the relation is satisfied, we say that $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are compatible with each other (*i.e.,* they can potentially be both inliers); however, if the relation is *not* satisfied, then one of the measurements *must* be an outlier. Generalizing this example, we obtain the following general definition of *compatibility test*.

Definition 4 (Compatibility Test). *Given a subset of n measurements and the corresponding n-invariant, a compatibility test is a binary condition (computed using the invariant), such that if the condition fails, then the set of measurements* must *contain at least one outlier.*

Note that we require the test to be *sound* (*i.e.,* it does not detect outliers when testing a set of inliers), but may not be *complete* (*i.e.,* the test might pass even in the presence of outliers). This property is important since our goal is to prune as many outliers as we can, while preserving the inliers. Also note that the test detects if the set contains outliers, but does not provide information on *which* measurements are outliers. We are going to fill in this gap below.

C. From Compatibility Tests to Compatibility Hypergraph

For a problem with an n-invariant, we describe the results of the compatibility tests for all subsets of n measurements using a *compatibility hypergraph*.

Definition 5 (Compatibility Hypergraph). *Given a compatibility test with n measurements, define the compatibility hypergraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as an n-uniform undirected hypergraph,*[3] *where each node v in the node set $\mathcal{V}$ is associated to a measurement in (4), and an hyperedge e (connecting a subset of n measurements) belongs to the edge set $\mathcal{E}$ if and only if its subset of measurements passes the compatibility tests.*

[3] In an n-uniform hypergraph, each hyperedge involves exactly n nodes.

Algorithm 1: ROBIN

Input: set of measurements $\mathcal{Y}$ and model (4); n-invariant functions $(\boldsymbol{f}, \boldsymbol{F})$ (for some n); inlier noise bound β;
Output: subset $\mathcal{Y}^\star \subset \mathcal{Y}$;
% Initialize compatibility graph
$\mathcal{V} = \mathcal{Y}$; % each node is a measurement
$\mathcal{E} = \emptyset$; % start with empty hyperedge set
% Perform compatibility tests
for all subsets $\mathcal{M} \subset \mathcal{Y}$ **of size** n **do**
  if `testCompatibility`$(\mathcal{M}, \boldsymbol{f}, \boldsymbol{F})$ = pass **then**
    add a hyperedge $e = \mathcal{M}$ to $\mathcal{E}$;
  end
end
% Find compatible measurements
$\mathcal{Y}^\star =$ `max_hyperclique`$(\mathcal{V}, \mathcal{E})$;
return: $\mathcal{Y}^\star$.

Note that in the case where $n = 2$, the above definition reduces to a regular undirected graph. Building the compatibility graph requires looping over all subsets of n measurements and, whenever the subset passes the compatibility test, adding a hyperedge between the corresponding n nodes in the graph. Note that these checks are very fast and easy to parallelize since they only involve checking boolean conditions (*e.g.,* (12)) without computing an estimate (as opposed to RANSAC).

Inlier Structures in Compatibility Hypergraphs. Here we show that the inliers in the set of measurements (4) are contained in a single hyperclique of the compatibility hypergraph. Let us start with some definitions.

Definition 6 (Hypercliques and Maximum Hyperclique in Hypergraphs). *A hyperclique of an n-uniform hypergraph $\mathcal{G}$ is a set of vertices such that any subsets of n vertices is connected by an hyperedge in $\mathcal{G}$. The* maximum *hyperclique of $\mathcal{G}$ is the hyperclique with the largest number of vertices.*

Again, in the case where $n = 2$, the above definition reduces to the usual clique and maximum clique definition. Given a compatibility hypergraph $\mathcal{G}$, the following result relates the set of inliers in (4) with hypercliques in $\mathcal{G}$ (proof in Appendix A).

Theorem 7 (Inliers and Hypercliques). *Assume we are given measurements (4) (with inlier noise bound β) and the corresponding n-invariants; call $\mathcal{G}$ the corresponding compatibility hypergraph. Then, assuming there are at least n inliers, all inliers belong to a single hyperclique in $\mathcal{G}$.*

Theorem 7 implies that we can look for inliers by computing hypercliques in the compatibility graph. Since we expect to have more compatible inliers than outliers, here we propose to compute the maximum hyperclique; this approach is shown to work extremely well in practice in Section VIII. Appendix B provides a longer discussion on how this approach compares with our original proposal in [90]. Appendix C describes an algorithm to find the maximum hyperclique.

D. ROBIN: Graph-theoretic Outlier Rejection

This section summarizes all the findings above into a single algorithm for graph-theoretic outlier pruning, named ROBIN (*Reject Outliers Based on INvariants*). ROBIN's pseudocode is given in Algorithm 1. The algorithm takes a set of measurements $\mathcal{Y}$ in input, and outputs a subset $\mathcal{Y}^\star \subset \mathcal{Y}$ from

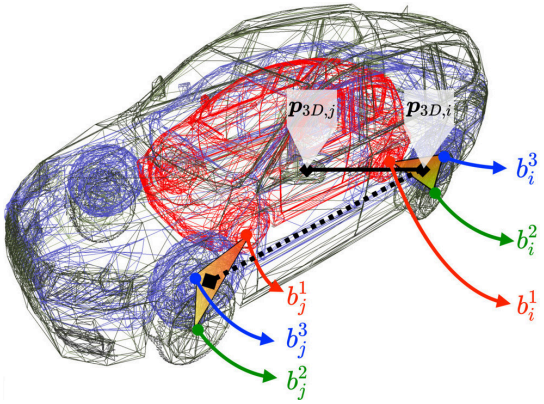


Fig. 4. Example of compatibility test with 3 CAD models of cars (red, dark green, blue, indexed from 1 to 3). (Noiseless) inliers (*e.g.,* the detection of the back wheel $\boldsymbol{p}_{3D,i}$ in the figure) must fall in the convex hull of the corresponding points on the CAD models (*e.g.,* triangle $\boldsymbol{b}_i^1 - \boldsymbol{b}_i^2 - \boldsymbol{b}_i^3$ encompassing the back wheel positions across CAD models). This restricts the relative distance between two inliers and allows filtering out outliers. For instance, the dashed black line shows a distance that is compatible with the location of the convex hulls, while the solid black line is too short compared to the relative position of the wheels (for any car model) and allows pointing out that there is an outlier (*i.e.,* $\boldsymbol{p}_{3D,j}$ in the figure).

which many outliers have been pruned. If the problem admits an n-measurement invariant, ROBIN first performs compatibility tests on all subsets of n measurements and builds the corresponding compatibility hypergraph (lines 3-11). Then, it uses a maximum hyperclique solver to compute and return the subset of measurements surviving outlier pruning (lines 12-14). We have implemented ROBIN in C++ and Python, using CVXPY [22] to solve (MILP); in the special case where $n = 2$, *i.e.,* the compatibility graph is an ordinary graph, we use the parallel maximum clique solver from [82]. We remark that ROBIN is not guaranteed to reject all outliers, *i.e.,* some outliers may still pass the compatibility tests and end up in the maximum hyperclique. Indeed, as mentioned in the introduction, ROBIN is designed to be a preprocessing step to prune gross outliers and enhance the robustness of existing robust estimators. Note that Algorithm 1 can be applied to various estimation problems, as long as suitable compatibility tests are defined (*i.e.,* `testCompatibility` function). In [90], we report applications to many geometric perception problems, including point cloud registration, point-with-normal registration, and camera pose estimation. In the following section, we develop compatibility tests for 3D-3D and 2D-3D category-level perception problems.

V. Stage 1 (continued): Application to Category-Level Perception

This section tailors ROBIN to category-level perception. Specifically, we develop *compatibility tests* for Problem 1 (Section V-A) and Problem 2 (Section V-B).

A. 3D-3D Category-level Compatibility Test

We develop a 3D-3D category-level compatibility test to check if a pair of 3D keypoint measurements are mutually compatible; our test generalizes results on instance-level perception, where the object shape is known [121], [27], [90].

We derive a pairwise invariant (*i.e.,* a 2-invariant) according to Definition 2. The challenge is to develop a set-valued function $\boldsymbol{F}$ that does not depend on the pose and shape parameters, which are unknown. Towards this goal, we show how to manipulate (2) to obtain a function $\boldsymbol{F}$ that does not depend on $\boldsymbol{R}$, $\boldsymbol{t}$, and $\boldsymbol{c}$. Taking the difference between measurement i and j in (2) leads to:

$$\boldsymbol{p}_{3D,j} - \boldsymbol{p}_{3D,i} = \boldsymbol{R}\textstyle\sum_{k=1}^{K} c^k(\boldsymbol{b}_j^k - \boldsymbol{b}_i^k) + (\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i})$$

where the translation cancels out in the subtraction. Now taking the ℓ_2 norm of both members we obtain:

$$\|\boldsymbol{p}_{3D,j} - \boldsymbol{p}_{3D,i}\| = \|\boldsymbol{R}\textstyle\sum_{k=1}^{K} c^k(\boldsymbol{b}_j^k - \boldsymbol{b}_i^k) + (\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i})\|$$

Using the triangle inequality, we have

$$\begin{aligned} -\|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\| \leq \|\boldsymbol{p}_{3D,j} - \boldsymbol{p}_{3D,i}\| - \left\|\boldsymbol{R}\textstyle\sum_{k=1}^{K} c_k(\boldsymbol{b}_j^k - \boldsymbol{b}_i^k)\right\| \\ \leq \|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\| \end{aligned} \tag{15}$$

Now observing that the ℓ_2 norm is invariant to rotation and rearranging the terms:

$$\begin{aligned} \left\|\textstyle\sum_{k=1}^{K} c^k(\boldsymbol{b}_j^k - \boldsymbol{b}_i^k)\right\| - \|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\| \leq \|\boldsymbol{p}_{3D,j} - \boldsymbol{p}_{3D,i}\| \\ \leq \left\|\textstyle\sum_{k=1}^{K} c^k(\boldsymbol{b}_j^k - \boldsymbol{b}_i^k)\right\| + \|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\| \end{aligned} \tag{16}$$

Taking the extreme cases over the possible shape coefficients:

$$\begin{aligned} & \overbrace{\min_{\boldsymbol{c}\geq 0, \mathbf{1}^\mathsf{T}\boldsymbol{c}=1} \left\|\textstyle\sum_{k=1}^{K} c^k(\boldsymbol{b}_j^k - \boldsymbol{b}_i^k)\right\|}^{b_{ij}^{\min}} - \|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\| \\ & \leq \|\boldsymbol{p}_{3D,j} - \boldsymbol{p}_{3D,i}\| \\ & \leq \underbrace{\max_{\boldsymbol{c}\geq 0, \mathbf{1}^\mathsf{T}\boldsymbol{c}=1} \left\|\textstyle\sum_{k=1}^{K} c^k(\boldsymbol{b}_j^k - \boldsymbol{b}_i^k)\right\|}_{b_{ij}^{\max}} + \|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\| \end{aligned} \tag{17}$$

Since $\sum_{k=1}^{K} c^k \boldsymbol{b}_j^k$ is a convex combinations of the points $\boldsymbol{b}_j^k$ $(k = 1\ldots, K)$ and hence lies in the convex hull of such points, the term $\|\sum_{k=1}^{K} c^k(\boldsymbol{b}_j^k - \boldsymbol{b}_i^k)\|$ represents the distance between two (unknown) points in the two convex hulls defined by the set of points $\boldsymbol{b}_j^k$ and $\boldsymbol{b}_i^k$ $(k = 1\ldots, K)$ (Fig. 4). The minimum $b_{ij}^{\min}$ and the maximum $b_{ij}^{\max}$ over the convex hulls can be easily computed, either in closed form or via small convex programs (see details in Appendix D). Accordingly,

$$\begin{aligned} & \|\boldsymbol{p}_{3D,j} - \boldsymbol{p}_{3D,i}\| \\ & \quad \in \left[b_{ij}^{\min} - \|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\|, b_{ij}^{\max} + \|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\|\right] \end{aligned} \tag{18}$$

Note that $b_{ij}^{\min}$ and $b_{ij}^{\max}$ only depend on the given library, and are independent on $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$. Therefore, they can be pre-computed. We can now define the pairwise invariant for Problem 1 with generative model defined in eq. (2):

Proposition 8 (3D-3D Category-level Pairwise Invariant and Compatibility Test). *Assume bounded noise* $\|\boldsymbol{\epsilon}_{3D,i}\| \leq \beta_{3D}$ *for* $i = 1, \ldots, N$. *The function* $\boldsymbol{f}_{3D}(\boldsymbol{p}_{3D,i}, \boldsymbol{p}_{3D,j}) \doteq \|\boldsymbol{p}_{3D,j} - \boldsymbol{p}_{3D,i}\|$ *is a pairwise invariant for eq. (2), with*

$$\boldsymbol{F}_{3D}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, \beta_{3D}) = \left[b_{ij}^{\min} - 2\beta_{3D}, b_{ij}^{\max} + 2\beta_{3D}\right]$$

where $\boldsymbol{\theta}_i = \{\boldsymbol{b}_i^k \,|\, k = 1, \ldots, K\}$ *and* $\boldsymbol{\theta}_j = \{\boldsymbol{b}_j^k \,|\, k = 1, \ldots, K\}$. *Therefore, two measurements* $\boldsymbol{p}_{3D,i}$ *and* $\boldsymbol{p}_{3D,j}$ *are compatible if* $\boldsymbol{f}_{3D}(\boldsymbol{p}_{3D,i}, \boldsymbol{p}_{3D,j}) \in \boldsymbol{F}_{3D}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, \beta_{3D})$.

The proof of the proposition trivially follows from eq. (18) and from the observation that $\|\boldsymbol{\epsilon}_{3D,i}\| \leq \beta_{3D}$ and $\|\boldsymbol{\epsilon}_{3D,j}\| \leq \beta_{3D}$ imply $\|\boldsymbol{\epsilon}_{3D,j} - \boldsymbol{\epsilon}_{3D,i}\| \leq 2\beta_{3D}$.

Proposition 8 provides a compatibility test according to Definition 4. In words, a pair of measurements is mutually compatible if their distance $\|\boldsymbol{p}_{3D,j} - \boldsymbol{p}_{3D,i}\|$ matches the corresponding distances in the CAD models (lower-bounded by $b_{ij}^{\min}$ and upper-bounded by $b_{ij}^{\max}$) up to measurement noise β_{3D}. If a pair of measurements fails the compatibility test, then one of them must be an outlier. A geometric interpretation of the compatibility test (for $\beta_{3D} = 0$) is given in Fig. 4.

B. 2D-3D Category-level Compatibility Test

The pairwise invariant presented in the previous section was inspired by the fact that the distance between pairs of points is (a noiseless) invariant to rigid transformations. When it comes to 2D-3D problems, it is known that there is no (noiseless) invariant for 3D points in generic configurations under perspective projection [65]. One option would be to use invariants for special configurations of points, such as cross ratios for collinear points [90]; however, this would not be generally applicable to our problem, where 3D keypoints are arbitrarily distributed on the CAD models. Here, we take an alternative route and we directly design a *generalized* 3-invariant for generic 3D point projections.

Our 2D-3D category-level compatibility test draws inspiration from back-face culling in computer graphics [24]. The key idea is that when observing an object, the *winding order* of triplet of keypoints seen in the image (roughly speaking: if the points are arranged in clockwise or counterclockwise order) must be consistent with the arrangement of the corresponding triplet of keypoints in the CAD models. Therefore, we formulate a test by checking whether the observed winding order matches our expectation from the CAD models. In this section, we first define the notion of 2D and 3D winding orders, as well as the visibility and covisibility regions (camera locations where triplets of points are covisible); then we introduce a 3-invariant involving winding orders; finally, we combine winding orders and covisibility regions to develop a 2D-3D category-level compatibility test.

2D Winding Orders.

Winding orders refer to whether an ordered triplet of projected points are arranged in a clockwise or counterclockwise order. For example, if we enumerate the points in Fig. 5 in ascending order of their indices (*i.e.,* 1, 2, 3), camera 1 sees points in counterclockwise order, while camera 2 sees them in clockwise order.

Definition 9 (2D Winding Order). *Given three 2D image points $\boldsymbol{p}_{2D,i}$, $\boldsymbol{p}_{2D,j}$, and $\boldsymbol{p}_{2D,m}$ where $i < j < m$, their winding order is the orientation {clockwise, counterclockwise} of points when enumerating them in the order $i \to j \to m$.*

The following proposition allows computing the 2D winding order algebraically (proof in Appendix E).

Proposition 10 (2D Winding Order Computation). *Assume three non-collinear 2D image points $\boldsymbol{p}_{2D,i}$, $\boldsymbol{p}_{2D,j}$, and $\boldsymbol{p}_{2D,m}$,*

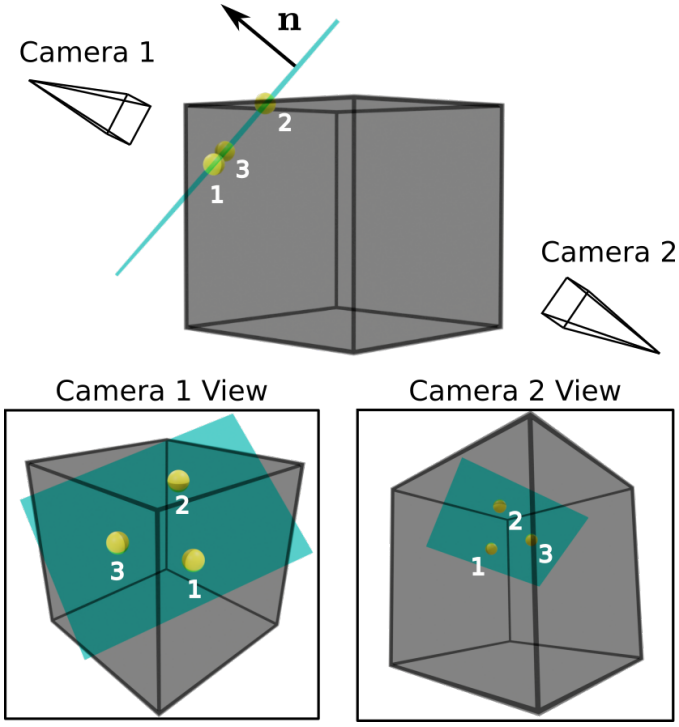


Fig. 5. Illustration of the concept of winding orders on a cube with three keypoints (in yellow) on its surface. The cube is opaque, but shown as semi-transparent for visualization. The plane cutting through all three keypoints (blue) has its normal $\boldsymbol{n}$ pointing outwards from the cube, following eq. (20). Camera 1 is in front of the plane, and the observed 2D winding order (Camera 1 View) is counterclockwise (enumerating in the order $1 \to 2 \to 3$), following the right-hand rule with thumb pointing towards the viewer. Camera 2 is behind the plane, and would observe a clockwise winding order. The only feasible winding order is counterclockwise as observed by Camera 1, since the keypoints would be occluded by the cube in Camera 2.

then the winding order W can be computed as:

$$W = \begin{cases} \textit{clockwise} & \textit{if } V > 0 \\ \textit{counterclockwise} & \textit{otherwise} \end{cases} \tag{19}$$

where $V = \det\left(\begin{bmatrix}\boldsymbol{p}_{2D,j} - \boldsymbol{p}_{2D,i} & \boldsymbol{p}_{2D,m} - \boldsymbol{p}_{2D,i}\end{bmatrix}\right)$ and $\det(\cdot)$ denotes the matrix determinant.

In words, Proposition 10 states that the 2D winding order can be calculated from the signed area of the parallelogram formed by $\boldsymbol{p}_{2D,j} - \boldsymbol{p}_{2D,i}$ and $\boldsymbol{p}_{2D,m} - \boldsymbol{p}_{2D,i}$.

Half-spaces and 3D Winding Orders. While Proposition 10 provides a simple way to compute the winding order for a triplet of 2D image points, towards our 2D-3D category-level invariant, we need to establish a notion of winding order also for the 3D shape keypoints on a CAD model. In the following we show that the winding order for a triplet of 3D points can be uniquely determined given the location of the camera. To develop some intuition, consider Fig. 5, where we have three 3D keypoints on the faces of a cube. The plane passing across the triplet of 3D keypoints divides the space into two half-spaces. Theorem 11 below shows that, whenever the camera lies within one of the half-spaces, only one winding order is possible. Therefore, if the keypoints are only covisible by camera locations in one of the half-spaces, their winding order is uniquely determined.

Let us formally define the half-spaces induced by the triplet plane. Let $\boldsymbol{b}_i^k$, $\boldsymbol{b}_j^k$, and $\boldsymbol{b}_m^k$ $(i < j < m)$ be three model keypoints on the k-th CAD model. Define the triplet normal

vector in the model's frame as follows (*cf.* $\boldsymbol{n}$ in Fig. 5):

$$\boldsymbol{n}^k_{i,j,m} = (\boldsymbol{b}^k_j - \boldsymbol{b}^k_i) \times (\boldsymbol{b}^k_m - \boldsymbol{b}^k_i) \tag{20}$$

So the triplet plane equation is: $(\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m} = 0$ for any $\boldsymbol{o} \in \mathbb{R}^3$. If $(\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m} > 0$ (resp. $(\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m} < 0$), $\boldsymbol{o}$ lies in the positive (resp. negative) half-space. In this section, one can think about $\boldsymbol{o}$ as the optical center of the camera in the CAD model's frame, hence the inequalities above capture which half-space the camera is located in.

The following theorem connects winding order with the two half-spaces created by the triplet plane (proof in Appendix F).

Theorem 11 (Half-spaces and 3D Winding Orders). *Under perspective projection per eq. (3) with zero noise ($\boldsymbol{\epsilon}_{2D,i} = \boldsymbol{\epsilon}_{2D,j} = \boldsymbol{\epsilon}_{2D,m} = 0$), the following equality holds:*

$$\begin{aligned} &\operatorname{sgn}\left((\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m}\right) \\ &= -\operatorname{sgn}\left(\det\left[\boldsymbol{p}_{2D,j} - \boldsymbol{p}_{2D,i} \quad \boldsymbol{p}_{2D,m} - \boldsymbol{p}_{2D,i}\right]\right) \end{aligned} \tag{21}$$

where $\operatorname{sgn}(\cdot)$ *is the signum function.*

The theorem states that the half-space the camera is located in ($\operatorname{sgn}\left((\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m}\right)$) uniquely determines the 2D winding order of the projection of the 3D points. This is not informative if the camera can be anywhere, since both winding orders are possible. In the following, we use the idea of covisibility regions to restrict potential locations of the camera, such that we can associate a possibly unique winding order to triplets of 3D keypoints.

Visibility and Covisibility Regions. Visibility and covisibility regions describe the set of camera locations from which keypoints are visible. The 3D object observed by the camera is assumed to be opaque, hence a keypoint visible from one viewpoint, might not be visible from another, due to self-occlusions. Fig. 5 demonstrates this concept: due to self-occlusions, the three keypoints on the cube are visible to Camera 1, but occluded (*i.e.,* not visible) in Camera 2.

We now define the visibility region of keypoints on a shape, following the standard definition [39].

Definition 12 (Visibility Region). *The visibility region of a keypoint $\boldsymbol{b}^k_i$ is the set of 3D points $\{\boldsymbol{o}\}$ such that line segments connecting $\boldsymbol{o}$ and $\boldsymbol{b}^k_i$ do not intersect the k-th shape model.*

We also define covisibility regions for keypoints triplets, which are the 3D space in which all three keypoints are visible.

Definition 13 (Covisibility Region). *The covisibility region of a triplet of keypoints $\boldsymbol{b}^k_i$, $\boldsymbol{b}^k_j$, and $\boldsymbol{b}^k_m$ is the intersection of the visibility region of each keypoint.*

Fig. 6(b) shows visibility and covisibility regions surrounding the `Q7-SUV` model from the ApolloScape dataset for selected triplets of keypoints. For the triplet $(0, 1, 8)$, their covisibility regions are to the front of the car, whereas for $(12, 14, 17)$, their covisibility regions are to the left of the car. Notice how the relative positions between triplet half-spaces and covisibility regions affect the visible winding orders; see Fig. 6(c). The plane formed by $(0, 1, 8)$ cuts through their covisibility regions, hence both winding orders are visible. For $(12, 14, 17)$, their covisibility regions are on one side of the keypoints plane, so the only visible winding order is clockwise. This observation is formalized below.

Corollary 14 (Covisibility-constrained 3D Winding Orders). *The projection of a triplet of 3D keypoints $\boldsymbol{b}^k_i$, $\boldsymbol{b}^k_j$, and $\boldsymbol{b}^k_m$ is arranged in counterclockwise winding order (following Definition 9) if and only if*

$$\{\boldsymbol{o} \in \mathbb{R}^3 \mid (\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m} > 0, \boldsymbol{o} \in \mathcal{C}\} \neq \varnothing \tag{22}$$

where $\mathcal{C}$ is the covisibility region of $\boldsymbol{b}^k_i$, $\boldsymbol{b}^k_j$, and $\boldsymbol{b}^k_m$ on the k-th shape. Similarly, they can be viewed in clockwise winding order if and only if

$$\{\boldsymbol{o} \in \mathbb{R}^3 \mid (\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m} < 0, \boldsymbol{o} \in \mathcal{C}\} \neq \varnothing \tag{23}$$

This corollary follows directly from Proposition 10, Theorem 11, and Definition 13. Remarkably, the two feasibility problems in eqs. (22) and (23) and can be solved *a priori*, since they only depend on the triplets of 3D keypoints and their normal. For polyhedral shapes, the constraint $\boldsymbol{o} \in \mathcal{C}$ can be expressed via linear constraints; in addition, if we replace the strict inequalities in eqs. (22) and (23) with non-strict ones and replace zeros with a small positive constant, the problems can be solved via linear programming. For complex shapes, we can check the conditions in eqs. (22) and (23) by splitting the space around the CAD models into voxels (*i.e.,* we discretize the set of possible $\boldsymbol{o}$) and ray tracing the keypoint to check visibility (see Appendix H). In summary, given a CAD model and for each triplet of 3D keypoints, using eqs. (22) and (23) we are able to predict if a covisible triplet will produce clockwise or counterclockwise keypoint projections.

Definition 15 (Feasible Winding Orders Dictionary). *For shape k, its feasible winding orders dictionary $\mathcal{W}_k : \{1, \dots, N\}^3 \to \mathcal{P}(\{-1, +1\})$ (where $\mathcal{P}(\cdot)$ denotes the power set,* i.e., *$\{+1, -1, \pm 1, \emptyset\}$) is defined as*

$$\mathcal{W}_k(i, j, m) = \{\operatorname{sgn}\left((\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m}\right) \mid \forall \boldsymbol{o} \in \mathcal{C}\} \tag{24}$$

*where $\{\operatorname{sgn}\left((\boldsymbol{o} - \boldsymbol{b}^k_i) \cdot \boldsymbol{n}^k_{i,j,m}\right) \mid \forall \boldsymbol{o} \in \mathcal{C}\}$ is empty if both eq. (22) and eq. (23) are false (*i.e., *when the triplet is never covisible); it contains $+1$ if (22) is true; it contains -1 if (23) is true; it contains both $+1$ and -1 if both (22) and (23) are true.*

We are now ready to define our generalized 3-invariant and the corresponding compatibility test.

2D-3D Invariant and Compatibility Test. We solve the two feasibility problems (22) and (23) for all triplets and CAD models and obtain a *dictionary of feasible winding orders*. In essence, each dictionary serves as a compatibility check for a single shape: for observed keypoints $\boldsymbol{p}_{2D,i}$, $\boldsymbol{p}_{2D,j}$, and $\boldsymbol{p}_{2D,m}$, if the observed winding orders are not in $\mathcal{W}_k(i, j, m)$, then the triplets are not compatible. However, this dictionary is only for a single shape. To formulate a 2D-3D category-level invariant, we need to create a dictionary of feasible winding orders for all K shapes. We address this in Proposition 16 below.

Proposition 16 (2D-3D Category-level Invariant and Compatibility Test). *Assume the keypoints in eq. (3) are generated by one of the shapes $\{1, \dots, K\}$, that the reprojection noise is bounded by β (*i.e., *$\|\boldsymbol{\epsilon}_{2D,i}\| \leq \beta, \|\boldsymbol{\epsilon}_{2D,j}\| \leq \beta, \|\boldsymbol{\epsilon}_{2D,m}\| \leq \beta$), and that β is small enough for Theorem 11 to hold true; then*

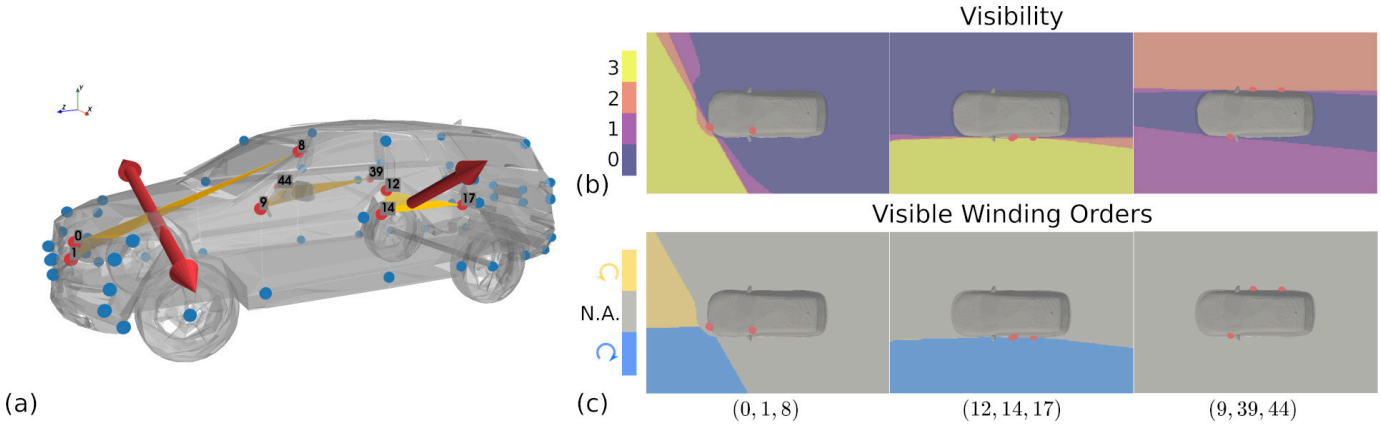


Fig. 6. Visualization of 3D winding orders of three triplets of keypoints on the model `Q7-SUV` from the ApolloScape dataset. (a) CAD model with keypoints. Red points represent the selected keypoint triplets, while yellow planes indicate the planes formed by the triplets. Arrows point towards the half-space in which the triplets are covisible. Triplet (0, 1, 8) can be viewed in both clockwise and counterclockwise winding order in their covisibility region. Triplet (12, 14, 17) can be only viewed in counterclockwise winding order. Triplet (9, 39, 44) cannot be viewed in either winding order, as their covisibility region is empty. (b-c) Winding orders and visibility of the triplets in a volume surrounding the car. The top row (b) shows visibility of the keypoints on a slice along the xz-plane. Color-coded values represent how many keypoints are visible in the triplet; covisibility regions are shaded in yellow. The bottom row (c) shows winding orders of the triplets in the covisibility region, with beige denoting counterclockwise, blue clockwise, and gray corresponding to cases where the covisibility region is empty. The visibility region and winding order plots are generated following the ray tracing method described in Appendix H.

the functions $(\boldsymbol{f}_{2D}, \boldsymbol{F}_{2D})$ *is a* 3*-invariant for eq.* (3)*, with*

$$\boldsymbol{f}_{2D}(\boldsymbol{p}_{2D,i}, \boldsymbol{p}_{2D,j}, \boldsymbol{p}_{2D,m}) \doteq \det\left(\begin{bmatrix}\boldsymbol{p}_{2D,j} - \boldsymbol{p}_{2D,i} & \boldsymbol{p}_{2D,m} - \boldsymbol{p}_{2D,i}\end{bmatrix}\right) \tag{25}$$

$$\boldsymbol{F}_{2D}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, \boldsymbol{\theta}_m) = \bigcup_{k=1}^{K} \mathcal{W}_k(i,j,m) \tag{26}$$

where $\boldsymbol{\theta}_\cdot = \{\boldsymbol{b}_\cdot^k \mid k = 1, \dots, K\}$ *and* $\mathcal{W}_k$ *is the winding order dictionary for shape* k*, as per Definition 15. Therefore, a triplet of measurements* $\boldsymbol{p}_{2D,i}, \boldsymbol{p}_{2D,j}$ *and* $\boldsymbol{p}_{2D,m}$ *is compatible if* $\boldsymbol{f}_{2D}(\boldsymbol{p}_{2D,i}, \boldsymbol{p}_{2D,j}, \boldsymbol{p}_{2D,m}) \in \boldsymbol{F}_{2D}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, \boldsymbol{\theta}_m)$.

Intuitively, the proposition states that the observed winding order must match at least one of the winding orders contained in the feasible winding orders dictionary of all shapes. The non-compatible triplets from Proposition 16 are points with noise so large that the measured winding orders become inconsistent with the models. Under technical conditions (discussed in Appendix G), Proposition 16 holds true even when keypoints are generated by convex combinations of $\boldsymbol{b}_i^k$.

VI. Stage II: Certifiably Optimal Solvers for Category-Level Perception

While Stage 1 serves the purpose of filtering out a large fraction of gross outliers (without even computing an estimate), Stage 2 aims to use the remaining measurements to estimate the pose and shape parameters. In this section, we develop *certifiably optimal* solvers for Problem 1 and 2 in the *outlier-free* case (*i.e.,* assuming that ROBIN removed all the outliers). In Section VII, we further improve robustness by incorporating a *graduated non-convexity* scheme to handle potential left-over outliers in the measurements after ROBIN.

A. Certifiably Optimal Solver for Outlier-free 3D-3D Category-Level Perception

We show how to solve Problem 1 in the outlier-free case, where the noise $\boldsymbol{\epsilon}_{3D,i}$ is assumed to follow a zero-mean Gaussian distribution. In the outlier-free case, a standard formulation for the pose and shape estimation problem leads to a *regularized non-linear least squares* problem:

$$\min_{\substack{\boldsymbol{R}\in SO(3),\\ \boldsymbol{t}\in\mathbb{R}^3, \mathbf{1}^T\boldsymbol{c}=1}} \sum_{i=1}^{N} w_i \left\| \boldsymbol{p}_{3D,i} - \boldsymbol{R}\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k - \boldsymbol{t} \right\|^2 + \lambda \|\boldsymbol{c}\|^2 \quad \text{(3D-3D)}$$

where the first summand in the objective minimizes the residual error w.r.t. the generative model (2) ($w_i \geq 0, i = 1, \dots, N$ are given weights), and the second term provides an ℓ_2 regularization (*a.k.a. Tikhonov regularization* [101]) of the shape coefficients $\boldsymbol{c}$ (controlled by the user-specified parameter $\lambda \geq 0$). Note that for mathematical convenience we replaced the constraint $\boldsymbol{c} \in \Delta_K$ with the constraint $\mathbf{1}^T\boldsymbol{c} = 1$, where $\mathbf{1}$ is a vector with all entries equal to 1; in other words, we force the shape coefficients to sum-up to 1 but allow them to be negative. Numerically, the regularization term ensures the problem is well-posed regardless of the number of shapes in the library (otherwise, the problem would be under-constrained when K is large). From the probabilistic standpoint, problem (3D-3D) is a *maximum a posteriori* estimator assuming that the keypoints measurement noise follows a zero-mean Gaussian with covariance $\frac{1}{w_i}\mathbf{I}_3$ (where $\mathbf{I}_3$ is the 3-by-3 identity matrix) and we have a zero-mean Gaussian prior with covariance $\frac{1}{\lambda}$ over the shape parameters $\boldsymbol{c}$ (see Appendix I). Problem (3D-3D) is non-convex due to the product between rotation $\boldsymbol{R}$ and shape parameters $\boldsymbol{c}$ in the objective, and due to the nonconvexity of the constraint set $SO(3)$ the rotation $\boldsymbol{R}$ is required to belong to, see *e.g.,* [81]. Therefore, existing approaches based on local search [56], [34], [77] are prone to converge to local minima corresponding to incorrect estimates.

3D-3D solver overview. We develop the first certifiably optimal algorithm to solve (3D-3D). Towards this goal we show that (i) the translation $\boldsymbol{t}$ in (3D-3D) can be solved in closed form given the rotation and shape parameters (Section VI-A1), (ii) the shape parameters $\boldsymbol{c}$ can be solved in closed form given the rotation (Section VI-A2), and (iii) the rotation can be estimated (independently on shape and translation) using a tight semidefinite relaxation (Section VI-A3).

1) Closed-form Translation Estimation: By inspection of (3D-3D), we observe the vector $\boldsymbol{t}$ is unconstrained and appears quadratically in the cost function. Therefore, for any choice of $\boldsymbol{R}$ and $\boldsymbol{c}$, the optimal translation can be computed in closed-form as:

$$\boldsymbol{t}^\star(\boldsymbol{R},\boldsymbol{c}) = \boldsymbol{y}_w - \boldsymbol{R}\textstyle\sum_{k=1}^K c_k \boldsymbol{b}_{k,w} \tag{27}$$

where

$$\boldsymbol{y}_w \triangleq \frac{1}{(\sum_{i=1}^N w_i)}\sum_{i=1}^N w_i \boldsymbol{p}_{3\text{D},i}, \quad \boldsymbol{b}_{k,w} \triangleq \frac{1}{(\sum_{i=1}^N w_i)}\sum_{i=1}^N w_i \boldsymbol{b}_i^k,$$

are the weighted centroids of $\boldsymbol{p}_{3\text{D},i}$ and $\boldsymbol{b}_i^k$'s. This manipulation is common in related work, *e.g.,* [125], [117].

2) Closed-form Shape Estimation: Substituting the optimal translation (27) (as a function of $\boldsymbol{R}$ and $\boldsymbol{c}$) back into (3D-3D), we obtain an optimization that only depends on $\boldsymbol{R}$ and $\boldsymbol{c}$:

$$\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3)\\ \mathbf{1}^\mathsf{T}\boldsymbol{c}=1}} \textstyle\sum_{i=1}^N \left\| \bar{\boldsymbol{y}}_i - \boldsymbol{R}\sum_{k=1}^K c_k \bar{\boldsymbol{b}}_i^k \right\|^2 + \lambda \|\boldsymbol{c}\|^2 \tag{28}$$

where

$$\bar{\boldsymbol{y}}_i \triangleq \sqrt{w_i}(\boldsymbol{p}_{3\text{D},i} - \boldsymbol{y}_w), \quad \bar{\boldsymbol{b}}_i^k \triangleq \sqrt{w_i}(\boldsymbol{b}_i^k - \boldsymbol{b}_{k,w}), \tag{29}$$

are the (weighted) relative positions of $\boldsymbol{p}_{3\text{D},i}$ and $\boldsymbol{b}_i^k$ w.r.t. their corresponding weighted centroids. Using the fact that the ℓ_2 norm is invariant to rotation, problem (28) is equivalent to:

$$\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3)\\ \mathbf{1}^\mathsf{T}\boldsymbol{c}=1}} \textstyle\sum_{i=1}^N \left\| \boldsymbol{R}^\mathsf{T}\bar{\boldsymbol{y}}_i - \sum_{k=1}^K c_k \bar{\boldsymbol{b}}_i^k \right\|^2 + \lambda \|\boldsymbol{c}\|^2 \tag{30}$$

We can further simplify the problem by adopting the following matrix notations:

$$\bar{\boldsymbol{y}} = \left(\bar{\boldsymbol{y}}_1^\mathsf{T}, \ldots, \bar{\boldsymbol{y}}_N^\mathsf{T}\right)^\mathsf{T} \in \mathbb{R}^{3N} \tag{31}$$

$$\bar{\boldsymbol{B}} = \begin{bmatrix} \bar{\boldsymbol{b}}_1^1 & \cdots & \bar{\boldsymbol{b}}_1^K \\ \vdots & \ddots & \vdots \\ \bar{\boldsymbol{b}}_N^1 & \cdots & \bar{\boldsymbol{b}}_N^K \end{bmatrix} \in \mathbb{R}^{3N\times K} \tag{32}$$

which allows rewriting (30) in the following compact form:

$$\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3)\\ \mathbf{1}^\mathsf{T}\boldsymbol{c}=1}} \left\| \bar{\boldsymbol{B}}\boldsymbol{c} - (\mathbf{I}_N \otimes \boldsymbol{R}^\mathsf{T})\bar{\boldsymbol{y}} \right\|^2 + \lambda \|\boldsymbol{c}\|^2 \tag{33}$$

Now the reader can again recognize that —for any choice of $\boldsymbol{R}$— problem (33) is a linearly-constrained linear least squares problem in $\boldsymbol{c}$, which admits a closed-form solution.

Proposition 17 (Optimal Shape). *For any choice of rotation $\boldsymbol{R}$, the optimal shape parameters that solve (33) can be computed in closed form as:*

$$\boldsymbol{c}^\star(\boldsymbol{R}) = 2\boldsymbol{G}\bar{\boldsymbol{B}}^\mathsf{T}(\mathbf{I}_N \otimes \boldsymbol{R}^\mathsf{T})\bar{\boldsymbol{y}} + \boldsymbol{g} \tag{34}$$

where we defined the following constant matrices and vectors:

$$\bar{\boldsymbol{H}} \triangleq 2(\bar{\boldsymbol{B}}^\mathsf{T}\bar{\boldsymbol{B}} + \lambda \mathbf{I}_K) \tag{35}$$

$$\boldsymbol{G} \triangleq \bar{\boldsymbol{H}}^{-1} - \frac{\bar{\boldsymbol{H}}^{-1}\mathbf{1}\mathbf{1}^\mathsf{T}\bar{\boldsymbol{H}}^{-1}}{\mathbf{1}^\mathsf{T}\bar{\boldsymbol{H}}^{-1}\mathbf{1}}, \quad \boldsymbol{g} \triangleq \frac{\bar{\boldsymbol{H}}^{-1}\mathbf{1}}{\mathbf{1}^\mathsf{T}\bar{\boldsymbol{H}}^{-1}\mathbf{1}} \tag{36}$$

3) Certifiably Optimal Rotation Estimation: Substituting the optimal shape parameters (34) (as a function of $\boldsymbol{R}$) back into (33), we obtain an optimization that only depends on $\boldsymbol{R}$:

$$\min_{\boldsymbol{R}\in\mathrm{SO}(3)} \left\| \boldsymbol{M}(\mathbf{I}_N \otimes \boldsymbol{R}^\mathsf{T})\bar{\boldsymbol{y}} + \boldsymbol{h} \right\|^2 \tag{37}$$

where the matrix $\boldsymbol{M} \in \mathbb{R}^{(3N+K)\times 3N}$ and vector $\boldsymbol{h} \in \mathbb{R}^{3N+K}$ are defined as:

$$\boldsymbol{M} \triangleq \begin{bmatrix} 2\bar{\boldsymbol{B}}\boldsymbol{G}\bar{\boldsymbol{B}}^\mathsf{T} - \mathbf{I}_{3N} \\ 2\sqrt{\lambda}\boldsymbol{G}\bar{\boldsymbol{B}}^\mathsf{T} \end{bmatrix} \qquad \boldsymbol{h} \triangleq \begin{bmatrix} \bar{\boldsymbol{B}}\boldsymbol{g} \\ \boldsymbol{g} \end{bmatrix} \tag{38}$$

Problem (37) is a quadratic optimization over the non-convex set $\mathrm{SO}(3)$. It is known that $\mathrm{SO}(3)$ can be described as a set of quadratic equality constraints, see *e.g.,* [106], [81] or [117, Lemma 5]. Therefore, we can succinctly rewrite (37) as a *quadratically constrained quadratic program* (QCQP).

Proposition 18 (Optimal Rotation). *The category-level rotation estimation problem (37) can be equivalently written as a* quadratically constrained quadratic program *(QCQP):*

$$\begin{aligned} \min_{\tilde{\boldsymbol{r}}\in\mathbb{R}^{10}} \quad & \tilde{\boldsymbol{r}}^\mathsf{T}\boldsymbol{Q}\tilde{\boldsymbol{r}} \\ \text{s.t.} \quad & \tilde{\boldsymbol{r}}^\mathsf{T}\boldsymbol{A}_i\tilde{\boldsymbol{r}} = 0, \forall i = 1,\ldots,15 \end{aligned} \tag{39}$$

where $\tilde{\boldsymbol{r}} \triangleq [1, \mathrm{vec}\,(\boldsymbol{R})^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{10}$ is a vector stacking all the entries of the unknown rotation $\boldsymbol{R}$ in (37) (with an additional unit element), $\boldsymbol{Q} \in \mathcal{S}^{10}$ is a symmetric constant matrix (expression given in Appendix J), and $\boldsymbol{A}_i \in \mathcal{S}^{10}, i = 1,\ldots,15$ are the constant matrices that define the quadratic constraints describing the set $\mathrm{SO}(3)$ *[117, Lemma 5].*

While a QCQP is still a non-convex problem, it admits a standard semidefinite relaxation, described below.

Corollary 19 (Shor's Semidefinite Relaxation). *The following semidefinite program (SDP) is a convex relaxation of (39):*

$$\begin{aligned} \min_{\boldsymbol{X}\in\mathcal{S}^{10}} \quad & \mathrm{tr}\,(\boldsymbol{Q}\boldsymbol{X}) \\ \text{s.t.} \quad & \mathrm{tr}\,(\boldsymbol{A}_0\boldsymbol{X}) = 1, \\ & \mathrm{tr}\,(\boldsymbol{A}_i\boldsymbol{X}) = 0, \forall i = 1,\ldots,15 \\ & \boldsymbol{X} \succeq 0 \end{aligned} \tag{40}$$

Moreover, when the optimal solution $\boldsymbol{X}^\star$ of (40) has rank 1, it can be factorized as $\boldsymbol{X}^\star = \begin{bmatrix} 1 \\ \mathrm{vec}\,(\boldsymbol{R}^\star) \end{bmatrix}[1 \ \mathrm{vec}\,(\boldsymbol{R}^\star)]$ where $\boldsymbol{R}^\star$ is the optimal rotation minimizing (37).

The relaxation entails solving a small SDP (10×10 matrix size, and 16 linear equality constraints), hence it can be solved in milliseconds using standard interior-point methods (*e.g.,* MOSEK [64]). Similar to related quadratic problems over $\mathrm{SO}(3)$ [81], [121], [116], [11], [28], the relaxation (40) empirically produces rank-1 —and hence *optimal*— solutions in common problems. Even when the relaxation is not tight, the relaxation allows computing an estimate and a bound on its suboptimality (see Appendix L). The proposed solution falls in the class of *certifiable algorithms* (see [6] and Appendix A in [121]), since it allows solving a hard (non-convex) problem efficiently and with provable a posteriori guarantees.

4) Summary: The results in this section suggest a simple algorithm to compute a certifiably optimal solution to the pose and shape estimation problem (3D-3D): (i) we first estimate the rotation $\boldsymbol{R}^\star$ using the semidefinite relaxation (40); (ii) we retrieve the optimal shape $\boldsymbol{c}^\star(\boldsymbol{R}^\star)$ given the optimal rotation using (34); (iii) we retrieve the optimal translation $\boldsymbol{t}^\star(\boldsymbol{R}^\star, \boldsymbol{c}^\star)$ using (27). We call the resulting algorithm PACE3D* (Pose and shApe estimation for 3D-3D Category-level pErception).

B. Certifiably Optimal Solver for Outlier-free 2D-3D Category-Level Perception

We now show how to solve Problem 2 in the outlier-free case, where the noise $\boldsymbol{\epsilon}_{2\text{D},i}$ in the generative model (3) follows a zero-mean isotropic Gaussian distribution. In such a case, the maximum a posteriori estimator for Problem 2 becomes:

$$\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3)\\ \boldsymbol{t}\in\mathbb{R}^3,\boldsymbol{c}\in\Delta_K}} \sum_{i=1}^{N} w_i \left\| \boldsymbol{p}_{2\text{D},i} - \pi\left(\boldsymbol{R}\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k + \boldsymbol{t}\right)\right\|^2 + \lambda \left\|\boldsymbol{c}\right\|^2 \tag{41}$$

where $\lambda\left\|\boldsymbol{c}\right\|^2$ with $\lambda \geq 0$ is an ℓ_2 regularization on the shape parameters, and w_i are given non-negative weights. Eq. (41) minimizes the *geometric reprojection* error and belongs to a class of optimization problems known as *fractional programming* because the objective in (41) is a sum of *rational* functions. Unfortunately, it is generally intractable to obtain a globally optimal solution for fractional programming [85]. In fact, even if $\boldsymbol{c}$ is known in (41), searching for the optimal $(\boldsymbol{R}, \boldsymbol{t})$ typically resorts to branch-and-bound [35], [68], which runs in worst-case exponential time.

To circumvent the difficulty of fractional programming caused by the geometric reprojection error, we adopt an *algebraic* reprojection error that minimizes the *point-to-line* distance between each 3D keypoint (*i.e.,* $\boldsymbol{R}\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k + \boldsymbol{t}$) and the *bearing vector* emanating from the camera center to the 2D keypoint (*i.e.,* the bearing vector $\boldsymbol{v}_i = \tilde{\boldsymbol{p}}_{2\text{D},i}/\|\tilde{\boldsymbol{p}}_{2\text{D},i}\|$, where $\tilde{\boldsymbol{p}}_{2\text{D},i} \doteq [\boldsymbol{p}_{2\text{D},i}^\mathsf{T}, 1]^\mathsf{T}$ is the homogenization of $\boldsymbol{p}_{2\text{D},i}$):

$$\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3)\\ \boldsymbol{t}\in\mathbb{R}^3,\boldsymbol{c}\in\Delta_K}} \sum_{i=1}^{N} w_i \left\| \boldsymbol{R}\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k + \boldsymbol{t}\right\|_{\boldsymbol{W}_i}^2 + \lambda \left\|\boldsymbol{c}\right\|^2 \tag{2D-3D}$$

where $\boldsymbol{W}_i := \mathbf{I}_3 - \boldsymbol{v}_i\boldsymbol{v}_i^\mathsf{T}$ and $\|\boldsymbol{p}\|_{\boldsymbol{W}_i}^2 := \boldsymbol{p}^\mathsf{T}\boldsymbol{W}_i\boldsymbol{p}$ computes the distance from a given 3D point $\boldsymbol{p}$ to a bearing vector $\boldsymbol{v}_i$. The point-to-line objective in (2D-3D) has been adopted in other works, including [87], [120]. Below, we omit the weights w_i, since they can be directly included in the matrix $\boldsymbol{W}_i$.

2D-3D solver overview. We develop the first certifiably optimal algorithm to solve problem (2D-3D). We show that (i) the translation $\boldsymbol{t}$ in (2D-3D) can be solved in closed form given the rotation and shape parameters (Section VI-B1), and (ii) the shape parameters $\boldsymbol{c}$ and the rotation $\boldsymbol{R}$ can be estimated using a tight semidefinite relaxation (Section VI-B2).

1) Closed-form Translation Estimation: Similar to Section VI-A1, our first step is to algebraically eliminate the translation $\boldsymbol{t}$ in (2D-3D). By deriving the gradient of the objective of (2D-3D) and setting it to zero, we obtain that

$$\boldsymbol{t}^\star = -\sum_{i=1}^{N} \widetilde{\boldsymbol{W}}_i \boldsymbol{R}\left(\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k\right) \tag{42}$$

where $\boldsymbol{W} \doteq \sum_{i=1}^{N} \boldsymbol{W}_i$ and $\widetilde{\boldsymbol{W}}_i \doteq \boldsymbol{W}^{-1}\boldsymbol{W}_i$. Inserting (42) back into (2D-3D), we get the following translation-free problem

$$\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3)\\ \boldsymbol{c}\in\Delta_K}} \sum_{i=1}^{N} \left\| \boldsymbol{R}\boldsymbol{s}_i(\boldsymbol{c}) - \sum_{j=1}^{N} \widetilde{\boldsymbol{W}}_j \boldsymbol{R}\boldsymbol{s}_j(\boldsymbol{c})\right\|_{\boldsymbol{W}_i}^2 + \lambda \left\|\boldsymbol{c}\right\|^2 \tag{43}$$

where we compactly wrote $\boldsymbol{s}_i(\boldsymbol{c}) = \sum_{k=1}^{K} c_k \boldsymbol{b}_i^k$.

2) Certifiably Optimal Shape and Rotation Estimation: In this case, it is not easy to decouple rotation and shape parameters as we did in Section VI-A2. Therefore, we adopt a more advanced machinery to globally optimize $\boldsymbol{R}$ and $\boldsymbol{c}$ together. Towards this goal, we observe that problem (43) is in the form of a *polynomial optimization problem* (POP), *i.e.,* an optimization problem where both objective and constraints can be written using polynomials:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^d} \quad & p(\boldsymbol{x}) \\ \text{s.t.} \quad & h_i(\boldsymbol{x}) = 0, i = 1, \ldots, l_h \\ & g_j(\boldsymbol{x}) \geq 0, j = 1, \ldots, l_g \end{aligned} \tag{44}$$

where $\boldsymbol{x} = [\mathrm{vec}\,(\boldsymbol{R})^\mathsf{T}, \boldsymbol{c}^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^d, d = K + 9$, denotes the vector of unknowns, $p(\boldsymbol{x})$ is a degree-4 objective polynomial corresponding to the objective in (43), $h_i(\boldsymbol{x}), i = 1, \ldots, l_h = 16$ are polynomial equality constraints (including $\boldsymbol{R} \in \mathrm{SO}(3)$ and $\mathbf{1}^\mathsf{T}\boldsymbol{c} = 1$), and $g_j(\boldsymbol{x}), j = 1, \ldots, l_g = K + 1$ are polynomial inequality constraints (*e.g.,* $c_k \geq 0$ for all k). $p(\boldsymbol{x})$ in (44) has degree four, which prevents us from applying Shor's semidefinite relaxation as in Corollary 19. However, writing (43) in the form (44) allows us to leverage a more general semidefinite relaxation technique called *Lasserre's hierarchy of moment and sum-of-squares relaxations* [49], [72] to solve (43) to certifiable global optimality.

Corollary 20 (Order-2 Lasserre's Moment Relaxation). *The following multi-block SDP is a convex relaxation of (39):*

$$\begin{aligned} \min_{\substack{\boldsymbol{X}=(\boldsymbol{X}_0,\boldsymbol{X}_1,\ldots,\boldsymbol{X}_{K+1})\\ \in\mathcal{S}^{n_0}\times\mathcal{S}^{n_1}\times\ldots\times\mathcal{S}^{n_{K+1}}}} \quad & \langle \boldsymbol{C}, \boldsymbol{X}\rangle \\ \textit{s.t.} \quad & \mathcal{A}(\boldsymbol{X}) = \boldsymbol{b}, \quad \boldsymbol{X} \succeq 0 \end{aligned} \tag{45}$$

where $\boldsymbol{X}_0$ is the moment matrix *of size $n_0 = \binom{K+11}{2}$, $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{K+1}$ are the so called* localizing matrices *(of size $n_i < n_0$ for $i \geq 1$) arising from the inequality constraints g_j in (44) and the additional constraint $\boldsymbol{c}^\mathsf{T}\boldsymbol{c} \leq 1$, $\boldsymbol{X} \succeq 0$ indicates each element of $\boldsymbol{X}$,* i.e., *$\boldsymbol{X}_i, i = 0, \ldots, K + 1$, is positive semidefinite, $\boldsymbol{C} = (\boldsymbol{C}_0, \boldsymbol{C}_1, \ldots, \boldsymbol{C}_{K+1})$ are known matrices with $\boldsymbol{C}_i = \mathbf{0}$ for $i \geq 1$, and $\mathcal{A}(\boldsymbol{X}) = \boldsymbol{b}$ collects all linear equality constraints on $\boldsymbol{X}$ (each scalar constraint is written as $\sum_{i=0}^{K+1} \langle \boldsymbol{A}_{ij}, \boldsymbol{X}_i\rangle = b_j$ for $j = 1, \ldots, m$).*

Moreover, when the optimal solution $\boldsymbol{X}^\star$ of (45) is such that $\boldsymbol{X}_0^\star$ has rank 1, then $\boldsymbol{X}_0^\star$ can be factorized as $\boldsymbol{X}_0^\star = [\boldsymbol{x}^\star][\boldsymbol{x}^\star]_2^\mathsf{T}$, where $\boldsymbol{x}^\star = [\mathrm{vec}\,(\boldsymbol{R}^\star)^\mathsf{T}, (\boldsymbol{c}^\star)^\mathsf{T}]^\mathsf{T}$ is a globally optimal for (43), and $[\boldsymbol{x}^\star]_2$ denotes the vector of monomials in the entries of $\boldsymbol{x}^\star$ of degree up to 2.

We refer the interested reader to [119, Section 2] for details about how to generate $(\mathcal{A}, \boldsymbol{b}, \boldsymbol{C})$ from the POP formulation (44). In practice, there exists an efficient Matlab implementation[4] that automatically generates the SDP relaxation (45)

[4] https://github.com/MIT-SPARK/CertifiablyRobustPerception

given a POP (44). The reader can observe that the SDP (45) is conceptually the same as the SDP (40) except that (45) has more than one positive semidefinite matrix decision variable. Appendix N contains more details about Lasserre's hierarchy optimality certificates and the rounding procedure to extract an estimate from the solution of the SDP (45).

3) Summary: We solve problem (2D-3D) by (i) solving the SDP (45) using MOSEK [64] and obtaining an estimate $(\widehat{\boldsymbol{R}}, \widehat{\boldsymbol{c}})$ and the corresponding suboptimality gap η, using the rounding procedure in Appendix N; then (ii) we compute $\hat{\boldsymbol{t}}$ from (42). If $\eta = 0$, we certify that $(\widehat{\boldsymbol{R}}, \hat{\boldsymbol{t}}, \widehat{\boldsymbol{c}})$ is a globally optimal solution to (2D-3D). We call this approach PACE2D* (*Pose and shApe estimation for 2D-3D Category-level pErception*).

VII. Stage II (continued): Further Robustness Through Graduated Non-Convexity

While in principle we can use our optimal solvers (PACE3D* and PACE2D*) directly after Stage 1, the measurements produced by ROBIN are potentially still contaminated by a few outliers. These outliers could still hinder the quality of the pose and shape estimates. Thus, the challenge lies in computing accurate estimates in the face of those remaining outliers.

Towards this goal, we add a robust loss function —in particular, a truncated least square loss— to problems (3D-3D) and (2D-3D), and solve the resulting optimization using a standard graduated non-convexity (GNC) [9], [115] approach. At each iteration, GNC alternates between solving a weighted least squares problem in the form (3D-3D) and (2D-3D) (these can be solved to certifiable optimality using PACE3D* and PACE2D*) and updating the weights for each measurement (which can be computed in closed form [115]). The interested reader can find more details in Appendix P.

VIII. Experiments

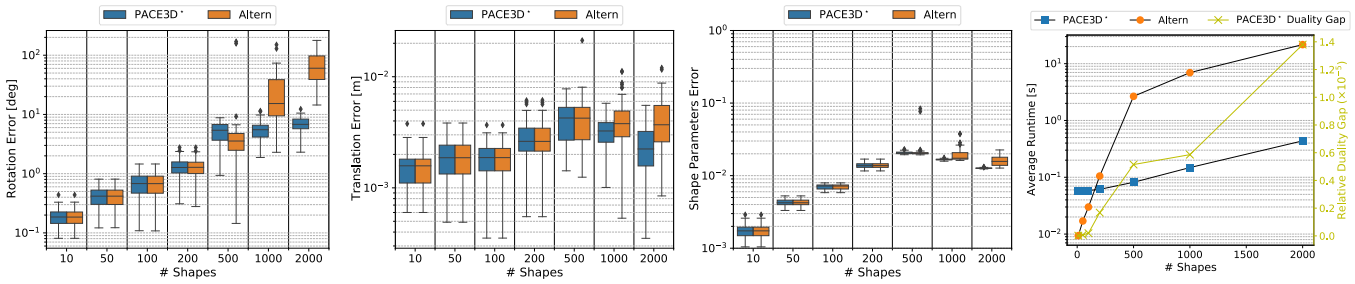
This section presents a comprehensive evaluations on our solvers. First, we showcase the optimality of PACE3D* and robustness of PACE3D# through experiments on synthetic data and the PASCAL3D+ dataset [114] (Section VIII-A). Then, we demonstrate the optimality of PACE2D* and robustness of PACE2D# through experiments on synthetic datasets (Section VIII-B). Finally, we test both PACE3D# and PACE2D# on ApolloScape, a self-driving dataset [110], [94]. Experiments with ApolloScape with PACE2D# are run on an AWS EC2 instance with 48 CPUs. All other experiments are run on a Linux computer with an Intel i9-9920X CPU at 3.5 GHz.

A. Optimality and Robustness of PACE3D* and PACE3D#

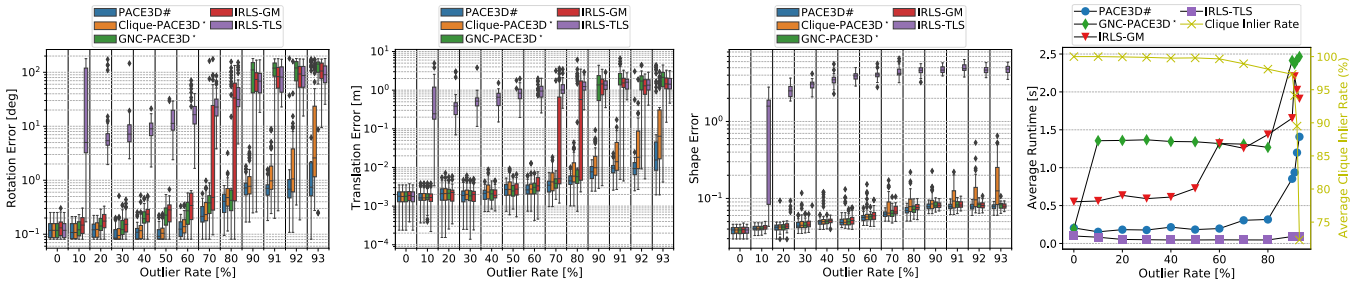
Optimality of PACE3D*. To evaluate the performance of PACE3D* in solving the outlier-free problem (3D-3D), we randomly simulate K shape models $\mathcal{B}_k$ whose points $\boldsymbol{b}_i^k$'s are drawn from an i.i.d. Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_3)$. We sample shape parameters $\boldsymbol{c}$ uniformly at random in $[0,1]^K$, and normalize $\boldsymbol{c}$ such that $\mathbf{1}^\mathsf{T}\boldsymbol{c} = 1$. Then we draw random poses $(\boldsymbol{R}, \boldsymbol{t})$ as in [121] and generate the measurements $\boldsymbol{p}_{3\mathrm{D},i}$ according to the model (2), where the noise $\boldsymbol{\epsilon}_{3\mathrm{D},i}$ follows $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_3)$ with standard deviation $\sigma = 0.01$. We fix $N = 100$, increase K from 10 up to 2000 and set the regularization factor $\lambda = \sqrt{K/N}$ so that larger regularization is imposed when K increases and the problem becomes more ill-posed. We compare PACE3D* with a baseline approach based on alternating minimization [56], [34], [77] (details in Appendix M) that offers no optimality guarantees (label: Altern).

Fig. 7(a) plots the statistics of rotation error (angular distance between estimated and ground-truth rotations), translation error, shape parameters error (ℓ_2 distance between estimated and ground-truth translation/shape parameters), as well as average runtime and relative duality gap (see also Appendix K for a formal definition). We make the following observations: (i) PACE3D* returns accurate pose and shape estimates up to $K = 2000$, while Altern starts failing at $K = 500$. (ii) Although Altern is faster than PACE3D* for small K (*e.g.,* $K < 200$), PACE3D* is orders of magnitude faster than Altern for large K. In fact, PACE3D* solves a fixed-size SDP regardless of K and the slight runtime increase is due to inversion of the dense matrix in (35)). (iii) The relaxation (40) is empirically tight (duality gap $< 10^{-4}$), certifying global optimality of the solution returned by PACE3D*.

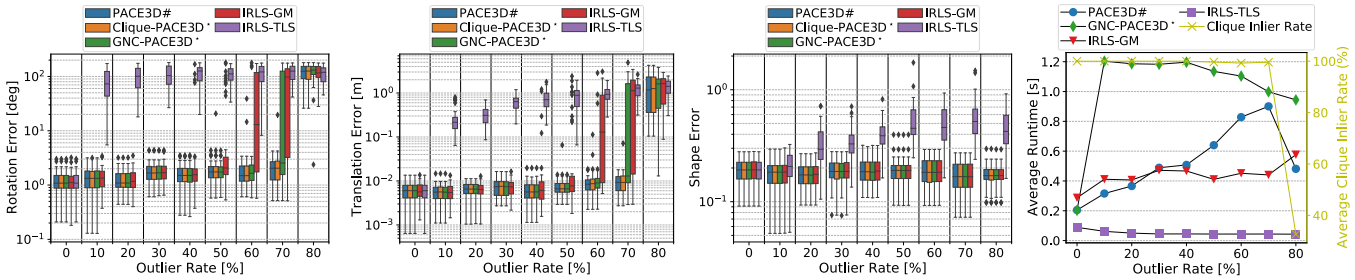
Robustness of PACE3D#. To test the robustness of PACE3D# on outlier-contaminated data, we follow the same data generation protocol as before, except that (i) when generating the CAD models, we follow a more realistic active shape model [19] where we first generate a mean shape $\mathcal{B}$ whose points $\boldsymbol{b}_i$'s are i.i.d. Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I}_3)$, and then each CAD model is generated from the mean shape by: $\boldsymbol{b}_i^k = \boldsymbol{b}_i + \boldsymbol{v}_i$, where $\boldsymbol{v}_i$ follows $\mathcal{N}(\mathbf{0}, r^2\mathbf{I}_3)$ and represents the *intra-class variation* of semantic keypoints with variation radius r; (ii) we replace a fraction of the measurements $\boldsymbol{p}_{3\mathrm{D},i}$ with arbitrary 3D points sampled according to $\mathcal{N}(\mathbf{0}, \mathbf{I}_3)$ and violating the generative model (2). We compare PACE3D# with two variants: Clique-PACE3D* (where, after pruning outliers using maximum clique, PACE3D* is applied *without* GNC) and GNC-PACE3D* (where GNC is applied *without* any outlier pruning), as well as two variants of the popular *iterative reweighted least squares* method: IRLS-TLS and IRLS-GM, where TLS and GM denote the truncated least squares cost function and the Geman-McClure cost function [100]. For fair comparison, we use PACE3D* inside PACE3D#, GNC-PACE3D*, IRLS-TLS, and IRLS-GM when updating $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$ given fixed weights. We set $\beta_{3D} = 0.05$ for outlier pruning and GNC. Fig. 7(b) plots the results under increasing outlier rates up to 93% when $N = 100$, $K = 10$, and $r = 0.1$. We make the following observations: (i) IRLS-TLS quickly fails (at 10% outlier rate) due to the highly non-convex nature of the TLS cost, while IRLS-GM is robust to 40% outliers. (ii) GNC-PACE3D* alone already outperforms IRLS-TLS and IRLS-GM and is robust to 60% outliers. (iii) With our maximum-clique outlier pruning, the robustness of PACE3D# is boosted to 92%, a level that is comparable to cases when the shapes are known (*e.g.,* [121]). In addition, outlier pruning speeds up the convergence of GNC-PACE3D* (*cf.* the runtime plot for GNC and PACE3D# in Fig. 7(b)). (iv) Even without GNC, the outlier pruning is so effective that PACE3D* alone is able to succeed with up to 90% outliers, albeit the estimates are typically less accurate than PACE3D#. In fact, looking at the clique inlier rate plot (yellow lineplot in Fig. 7(b)), the reader sees that the set of measurements after maximum clique pruning is almost free of outliers, explaining the surprising performance of Clique-PACE3D*. In Appendix Q-A, we show



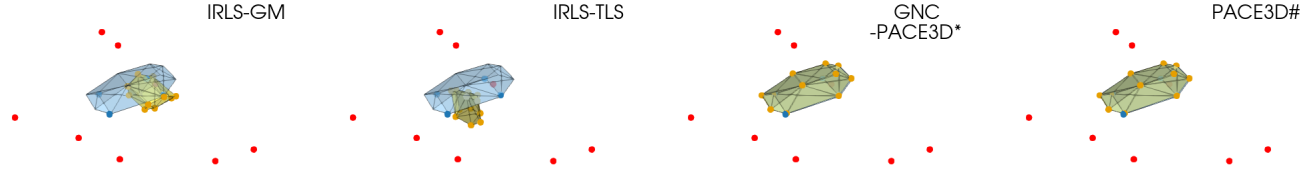
(a) Performance of PACE3D⋆ on outlier-free synthetic data: $N = 100$.



(b) Robustness of PACE3D# against increasing outliers on synthetic data: $N = 100$, $K = 10$, $r = 0.1$.



(c) Robustness of PACE3D# against increasing outliers on the *car* category in the PASCAL3D+ dataset [114]: $N = 12$, $K = 9$.



(d) Qualitative results of IRLS-GM, IRLS-TLS, GNC, and PACE3D# on a PASCAL3D+ instance with 70% outlier rate.

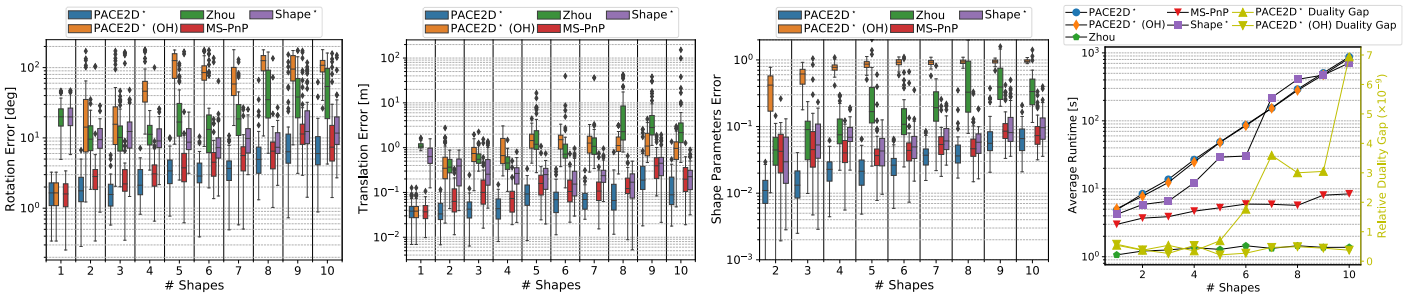
Fig. 7. Performance of PACE3D⋆ and PACE3D# compared with baselines in simulated experiments. (a) PACE3D⋆ compared with alternating minimization (Altern) on synthetic outlier-free data with $N = 100$ and K increasing from 10 to 2000; (b) PACE3D# and variants (Clique-PACE3D⋆ and GNC), compared with two variants of iterative reweighted least squares (IRLS-GM and IRLS-TLS) [100] on synthetic outlier-contaminated data with $N = 100$, $K = 10$, and outlier rates up to 93%; (c) same as (b) but using the *car* category CAD models from the PASCAL3D+ dataset [114], with $N = 12$, $K = 9$, and outlier rates up to 80%. Each boxplot and lineplot summarizes 50 Monte Carlo random runs. (d) Qualitative results of IRLS-GM, IRLS-TLS, GNC, and PACE3D# on a PASCAL3D+ instance with 70% outlier rate. Blue meshes represent the ground-truth shape, and yellow meshes represent the pose and shape estimated by each model. Red points represent outliers. In this case, both GNC and PACE3D# succeed, while IRLS-GM and IRLS-TLS failed.

extra results for $r = 0.2$ and $K = 50$, which further confirm PACE3D#'s robustness with up to 90% outliers.

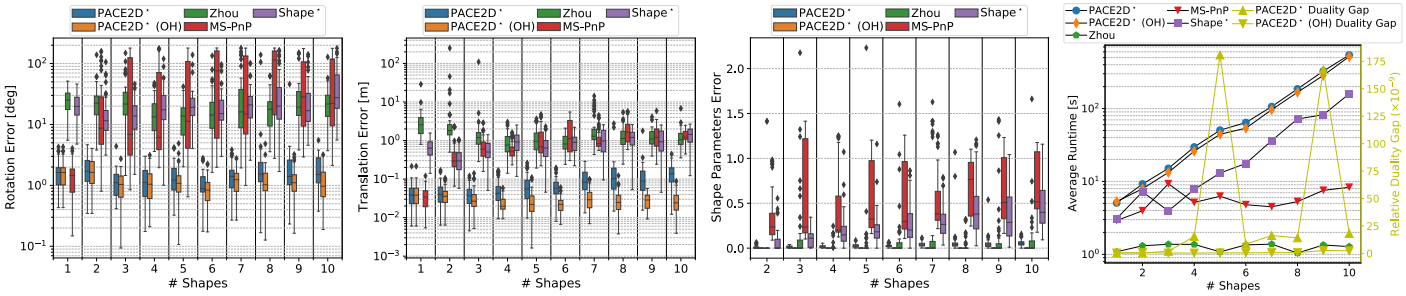
Robustness on PASCAL3D+. For a simulation setup that is closer to realistic scenarios, we use the CAD models from the *car* category in the PASCAL3D+ dataset [114], which contains $K = 9$ CAD models with $N = 12$ semantic keypoints. We randomly sample $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$ and add noise and outliers as before, and compare the performance of PACE3D# with other baselines, as shown in Fig. 7(c). The dominance of PACE3D# over other baselines, and the effectiveness of outlier pruning is clearly seen across the plots. PACE3D# is robust to 70% outliers (see Fig. 7(d) for a qualitative example), while other baselines break at a much lower outlier rate. Note that at 80% outlier rate, there are only two inlier semantic keypoints, making it pathological to estimate shape and pose.

B. *Optimality and Robustness of* PACE2D⋆ *and* PACE2D#

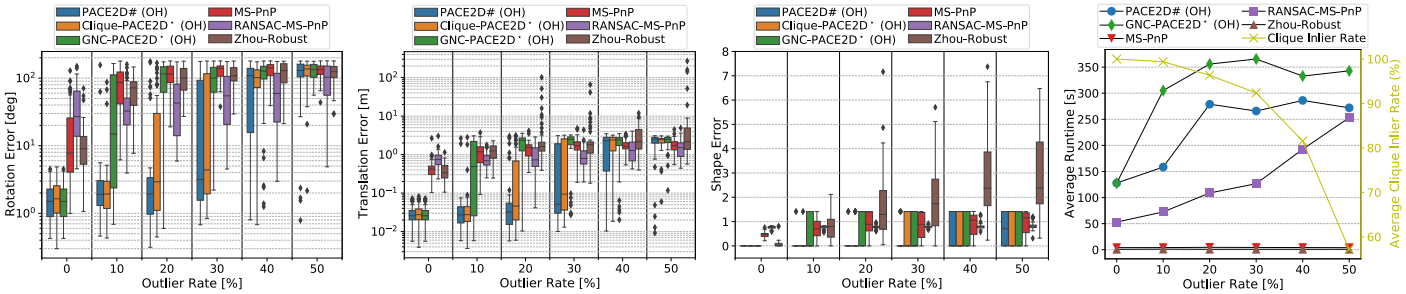
Optimality of PACE2D⋆. To evaluate the performance of PACE2D⋆ in solving the outlier-free problem (41), we randomly simulate K shapes $\mathcal{B}_k$ whose points $\boldsymbol{b}_i^k$'s are drawn from an i.i.d. Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \mathbf{I}_3)$. For the shape parameters $\boldsymbol{c}$, we use two different sampling processes: one where we sample $\boldsymbol{c}$ uniformly at random in the probability simplex Δ_K, and the other where we sample $\boldsymbol{c}$ as a one-hot vector. We draw random poses $(\boldsymbol{R}, \boldsymbol{t})$ such that the points lie in front of the camera, and generate the measurements $\boldsymbol{p}_{2D,i}$ according to the model (3), where the noise $\boldsymbol{\epsilon}_{2D,i}$ follows $\mathcal{N}(\boldsymbol{0}, \sigma^2 \mathbf{I}_2)$ with standard deviation $\sigma = 0.01$. We test both the proposed solver PACE2D⋆, as well as a variant that forces the shape vector to be a one-hot vector; the latter can be easily obtained by adding extra constraints in the POP (44) as discussed in Appendix O



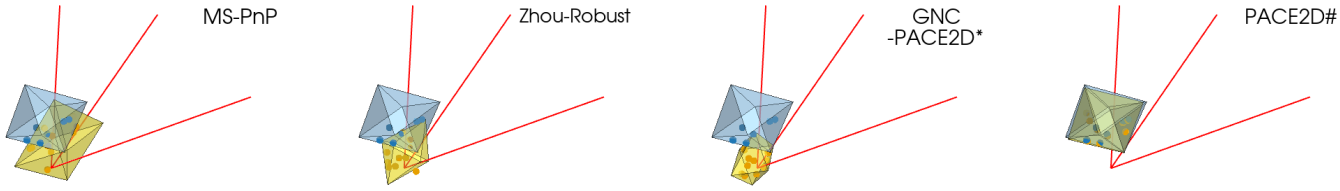
(a) Performance of PACE2D⋆(OH) and PACE2D⋆ on outlier-free synthetic data: $N = 8$; $\boldsymbol{c}$ sampled from Δ_K uniformly at random.



(b) Performance of PACE2D⋆(OH) and PACE2D⋆ on outlier-free synthetic data: $N = 8$; $\boldsymbol{c}$ sampled as a random one-hot vector.



(c) Robustness of PACE2D#(OH) against increasing outliers on synthetic data: $N = 10$, $K = 3$; $\boldsymbol{c}$ sampled as a random one-hot vector.



(d) Qualitative results of MS-PnP, Zhou-Robust, GNC-PACE2D⋆ and PACE2D# on a test instance with 30% outlier rate.

Fig. 8. Performance of PACE2D⋆(OH), PACE2D⋆, and PACE2D# compared with baselines in simulated experiments. (a) PACE2D⋆(OH) and PACE2D⋆ compared with MS-PnP, Zhou, and Shape⋆ on synthetic outlier-free data with varying number of shapes, where $\boldsymbol{c}$ is sampled uniformly at random from Δ_K. (b) same as (a) but with $\boldsymbol{c}$ being a random one-hot vector. (c) PACE2D# (OH) and variants compared with MS-PnP, RANSAC-MS-PnP and Zhou-Robust on synthetic outlier-contaminated data with varying outlier rates. (d) Qualitative results of MS-PnP, Zhou-Robust, GNC-PACE2D⋆, and PACE2D# on an instance with 30% outlier rate. Blue meshes represent the ground-truth shape, and yellow meshes represent the pose and shape estimated by each model. Red rays indicate outliers (bearing vectors originated from the camera center). In this case, PACE2D# succeeds while the other methods fail.

(label: PACE2D⋆(OH)). We compare both variants against (i) a baseline approach based on a local solver optimizing (41), starting from an initial guess obtained by running EPnP [52] on the mean shape (label: MS-PnP), (ii) a solver based on a convex relaxation using the weak perspective camera model [126] (label: Zhou), and (iii) a solver based on a tighter relaxation with the weak perspective model [117] (label: Shape⋆).

Fig. 8(a) and (b) respectively show the relevant statistics for the case where $\boldsymbol{c}$ is sampled uniformly at random from Δ_K (Case 1) and the case where $\boldsymbol{c}$ is a one-hot vector (Case 2). We report statistics for the rotation, translation, and shape errors. We also show the average runtime and relative duality gap. In Case 1 (Fig. 8(a)), we observe that PACE2D⋆ consistently outperforms all the other techniques. Not surprisingly, PACE2D⋆(OH) fails when $K \geq 2$, as it returns a one-hot $\boldsymbol{c}$, while the ground truth shape is a generic point in the probability simplex. While Zhou and Shape⋆ perform similarly at $K = 1$, Shape⋆ has lower errors at higher shape counts. Both Zhou and Shape⋆ perform significantly worse than PACE2D⋆, since the weak perspective projection model is only an approximation of the actual (fully perspective) camera geometry. In Case 2 (Fig. 8(b)), we observe that MS-PnP fails for $K \geq 2$, while both PACE2D⋆(OH) and PACE2D⋆ return accurate estimates for all K values, consistently outperforming all baselines. As expected, PACE2D⋆(OH) outperforms PACE2D⋆ in this case. Notably, PACE2D⋆(OH) returns the correct shape

estimate and achieves zero shape error at all K (see the shape error plot of Fig. 8(b), where we use a linear scale to better show zero errors). In both cases, the duality gap for PACE2D*(OH) and PACE2D* stays below 10^{-8}, indicating that the relaxation is empirically tight. Unfortunately, in terms of runtime, both PACE2D*(OH) and PACE2D* scale poorly with the number of shapes, leading to runtimes in the order of minutes already for a handful of shapes.

Robustness of PACE2D#. To test the robustness of PACE2D#, we use a different data generation procedure to enable the use of ROBIN. We first generate K octahedra, center-aligned at the origin, with vertices sampled component-wise in $[0.5, 3]$ m. The use of octahedra (convex shapes with known face planar equations) allows us to solve for sets of feasible winding orders —following Corollary 14— using linear programs (see Appendix H). We sample shape parameters $\boldsymbol{c}$ as random one-hot vectors. Then we draw random poses $(\boldsymbol{R}, \boldsymbol{t})$ such that the resulting camera locations lie on a sphere centered at the origin with radius of 5 m. For each camera location, we randomly sample $\boldsymbol{b}_i^k,\ i = 1 \dots N$ from each octahedron such that $\lfloor r \cdot N \rfloor$ of them lie on the weighted octahedron's visible faces where r is the outlier ratio. For the remaining $\boldsymbol{b}_i^k$, we sample them from the nonvisible faces of the octahedron. We generate the measurements $\boldsymbol{p}_{2\mathrm{D},i}$ according to eq. (3), where the noise $\boldsymbol{\epsilon}_{2\mathrm{D},i}$ follows $\mathcal{N}(\boldsymbol{0}, \sigma^2 \mathbf{I}_2)$ with $\sigma = 0.01$. For the $\boldsymbol{p}_{2\mathrm{D},i}$ generated by nonvisible $\boldsymbol{b}_i^k$'s, we replace their noise term $\boldsymbol{\epsilon}_{2\mathrm{D},i}$ with arbitrary 2D points violating the generative model (3). The regularization factor λ is set to 0.01.

Since the shape is generated as a one-hot vector, we use PACE2D# (OH) and consider two variants: GNC-PACE2D* (OH) (where GNC is applied to PACE2D*(OH) without ROBIN), and Clique-PACE2D* (where PACE2D* is applied after ROBIN without GNC). We also compare against Zhou-Robust, which is a robust version of Zhou's solver from [126], MS-PnP and RANSAC-MS-PnP where we wrap MS-PnP in a standard RANSAC loop, with the same inlier threshold as PACE2D# (OH). Fig. 8(c) plots the results under increasing outlier rates. PACE2D# (OH) remains robust up to 20%, while GNC-PACE2D* (OH), MS-PnP, RANSAC-MS-PnP, and Zhou-Robust already exhibit large median errors at 10% outlier rates. Interestingly, at around 10%, Clique-PACE2D* remains robust while GNC-PACE2D* (OH) starts to show severe failures. This remarks the effectiveness of ROBIN in filtering out outliers, also shown in the clique inlier rate plot in Fig. 8(c). Similar to PACE3D#, ROBIN improves the convergence rate of GNC; see runtime curves of PACE2D# (OH) and GNC-PACE2D* (OH) in Fig. 8 (c). Fig. 8(d) reports qualitative results on an simulated instance with 30% outlier rate; in this case, PACE2D# succeeds while the other methods fail. These results underline that 2D-3D category-level perception is a much harder problem compared to its 3D-3D counterpart. The proposed algorithms, while being competitive against baselines, are still slow and only robust to a small fraction of outliers. In certain cases, we have also observed RANSAC-MS-PnP to have a more graceful degradation for increasing outliers, see results in Appendix Q-B.

C. Vehicle Pose and Shape Estimation on ApolloScape

Setup and Baselines. We evaluate PACE3D# and PACE2D# on the ApolloScape dataset [110], [94]. The ApolloScape self-driving dataset is a large collection of multi-modal data collected in four different cities in China under varying lighting and road conditions [110]. For our experiments, we specifically use the subset of ApolloScape named ApolloCar3D. ApolloCar3D consists of high-resolution (3384×2710) images taken from the main ApolloScape dataset, with additional 2D annotations of semantic keypoints, ground truth poses, and 3D CAD models of car instances in each frame. The dataset contains a total of 5277 images, with an average of 11.7 cars per image, and a total of 79 ground-truth CAD models [94]. For each car, a total of 66 semantic keypoints were labeled on 2D images. In addition, the ground truth CAD model (out of the 79 models) is provided for each vehicle. Note this corresponds to having a one-hot vector for the ground-truth $\boldsymbol{c}$ in eqs. (2) and (3).

Fig. 9. We pass ground-truth annotated keypoints to ROBIN, using a winding order dictionary generated from ray tracing. Green dots represent keypoints inliers accepted by ROBIN. Red dots represent outliers rejected by ROBIN. In these four examples, ROBIN correctly rejects mislabelled keypoints. From top left clockwise: #8 and #49 are switched (with respect to the CAD models); #61 and #60 are switched; #34 is wrongly placed (should be #26 instead); #29 is wrongly placed (should be placed below the right tail light).

We compare PACE3D# and PACE2D# (OH) against DeepMANTA [15], 3D-RCNN [46], and GSNet [41], three recent state of the art methods for 3D vehicle pose estimation. We use the one-hot variant of PACE2D# because the ApolloScape ground truth shape coefficients are one-hot vectors. For our experiments, we use the official splits of the ApolloCar3D dataset. Namely, we use the validation split (200 images) for all the quantitative experiments shown below, consistent with the evaluation setups reported in other baseline methods.

We use the 2D semantic keypoints extracted by GSNet [41] as measurements for PACE2D#. We use the pretrained weights from [41] and reject keypoints with confidence less than 0.05. For PACE3D#, we additionally retrieve the corresponding depth from the depth images provided by ApolloScape for each 2D semantic keypoint; the resulting technique is labeled PACE3D#-ApolloDepths. For PACE2D#, we learn the dictionary of feasible winding orders for keypoints from the training set, as well as by performing ray tracing to keypoints in a volume surrounding each CAD model (see Appendix H).

Interestingly, when we applied ROBIN on the ground-truth annotations, we uncovered multiple mislabeled keypoints (see Fig. 9). This lends further credence to the effectiveness of ROBIN, and suggests ROBIN may also be helpful in terms of verifying datasets. While the 2D semantic keypoint an-

	A3DP-Rel ↑			A3DP-Abs ↑		
	mean	c-l	c-s	mean	c-l	c-s
DeepMANTA [15]	16.0	23.8	19.8	20.1	30.7	23.8
3D-RCNN [46]	10.8	17.8	11.9	16.4	29.7	19.8
GSNet [41]	20.2	**40.5**	19.9	18.9	**37.4**	18.4
PACE2D#	14.4	25.5	19.2	9.6	17.5	12.2
PACE3D#-ApolloDepths	**28.5**	37.4	**35.8**	**24.0**	36.4	**33.3**
PACE2D#-GTKeypoints	31.5	56.7	44.5	17.4	34.1	24.0
PACE2D#-GTReprojKeypoints	61.2	91.9	91.9	61.2	91.9	91.9
PACE3D#-GTDepths	37.3	44.9	43.5	36.5	43.6	42.6
PACE3D#-GTKeypoints	64.8	86.6	85.0	64.6	86.6	85.0

TABLE I
EVALUATION OF PACE# ON APOLLOSCAPE. RESULTS FOR DEEPMANTA,3D-RCNN, AND GSNET ARE TAKEN FROM [41]. THE BEST RESULT FOR EACH COLUMN IS HIGHLIGHTED IN BOLDFACE.

notations are provided by ApolloCar3D, the dataset does not provide the corresponding 3D keypoint annotations on the CAD models. To obtain the necessary 2D-3D correspondences, we manually labeled the 66 3D semantic keypoints on the 79 CAD models. We then provide them as the shape library to PACE3D#. For PACE2D#, we instead select 3 random models, including the ground truth model, as a shape library, since using the entire set leads to prohibitive runtime. We use $\lambda = 0.5$ and $\beta_{3D} = 0.15$ in PACE3D#, and $\lambda = 0.001$ and $\beta_{2D} = 0.01$ in PACE2D#.

Results. Table I shows the performance of PACE3D# and PACE2D# against various baselines (qualitative results can be found in Appendix Q-C). We use two metrics called A3DP-Rel and A3DP-Abs (for both, the higher the better) following the same definitions in [94]. They are measures of precision with thresholds jointly considering translation, rotation, and 3D shape similarity between estimated cars and ground truth. A3DP-Abs uses absolute translation thresholds, whereas A3DP-Rel uses relative ones. The *mean* column represents the average A3DP-Abs/Rel over 10 different thresholds. *c-l* represents a loose criterion (2.8 m for translation error, $\pi/6$ rad for rotation error, and 0.5 for shape similarity), and *c-s* represents a strict criterion (1.4 m for translation error, $\pi/12$ rad for rotation error, and 0.75 for shape similarity).

Overall, PACE2D# achieves performance comparable but slightly inferior to learning-based approaches in A3DP-Rel, while PACE3D# excels in both A3DP-Rel and A3DP-Abs. The main failure mode of PACE2D# is in its translation estimation: over 98% of the failures do not meet the translation threshold only. PACE3D# outperforms the baselines in terms of the *mean* and *c-s* criteria; this is partially expected since we use depth information, which is not available to the other methods at test time. In terms of the strict criterion *c-s*, PACE3D# outperforms competitors by a large amount, confirming that it can retrieve highly accurate estimates. When using the loose criterion *c-l*, GSNet is slightly better than PACE3D#, suggesting learning-based techniques may have more graceful failure modes.

Further Ablation. We also provide a final ablation study on PACE3D# and PACE2D#, to assess the impact of the keypoint quality. For PACE2D#, we test two variants: PACE2D#-GTKeypoints where ground-truth annotated 2D semantic keypoints are used, and PACE2D#-GTReprojKeypoints where projections of 3D ground-truth transformed keypoints are used, with occluded points removed. To assess the impact of depth and keypoint quality on PACE3D#, we test two variants: PACE3D#-GTDepths uses ground-truth depths obtained by ray-tracing the GSNet keypoints using ground-truth 3D car models, while PACE3D#-GTKeypoints uses ground-truth annotated 2D semantic keypoints with ground-truth depths.

GSNet Keypoint Detection	PACE3D#		PACE2D#	
	Max-clique	GNC	Max-clique	GNC
0.45 s	2 ms	0.45 s	5.88 s	535.75 s

TABLE II
TIMING BREAKDOWN FOR PACE2D# AND PACE3D#.

The results are reported in the bottom four rows of Table I. Both PACE2D#-GTReprojKeypoints and PACE2D#-GTKeypoints have better performance than PACE2D#, with PACE2D#-GTReprojKeypoints being significantly better. This indicates that (i) PACE2D# can achieve better performance with better keypoint detections, and (ii) ground-truth 2D keypoint annotations in ApolloScape are not entirely consistent with 3D reprojected keypoints. In terms of PACE3D#, we see PACE3D#-GTDepths outperforms baselines across all criteria, suggesting that if accurate depth measurements are available, PACE3D# can roughly double the performance of state-of-the-art methods in terms of *mean* and *c-s* criteria. PACE3D#-GTKeypoints shows the results produced by PACE3D# when using ground-truth keypoint detections and depths: this is the best potential accuracy PACE3D# could achieve if provided with perfect keypoint detections. In our tests, the average number of inliers produced by GSNet is 21.8%,[5] showing that there is still a large margin of improvement for state-of-the-art deep learning methods for semantic keypoint detection.

Runtime. Table II shows the timing breakdown for PACE3D# and PACE2D#. We also report the timing for the GSNet keypoint detection from [41] for completeness. For PACE3D#, the max-clique pruning is written in C++ and has negligible runtime, while GNC is implemented in Python. For PACE2D#, both the maximum hyperclique estimation and GNC are implemented in Python. While data in Table II for PACE2D# and PACE3D# are from different machines, the order of magnitude difference makes it clear that PACE3D# is significantly faster than PACE2D# thanks to PACE3D*'s compact semidefinite relaxation. While PACE2D# is impractically slow for real-world robotics applications, an optimized implementation of PACE3D# is amenable to practical applications.

IX. RELATED WORK

This section reviews related work on *category-level perception* and *outlier-robust estimation*.

A. Category-level Perception

Early approaches for category-level perception focus on 2D problems, where one has to locate objects —from human faces [69] to resistors [19]— in images. Classical approaches include *active contour models* [40], [16] and *active shape models* [19], [7]. These works use techniques like PCA to build a library of 2D landmarks from training data, and then use iterative optimization algorithms to estimate the 2D object locations in the images, rather than estimating 3D poses.

The landscape of category-level perception has been recently reshaped by the rapid adoption of **convolutional networks** [50], [45], [92]. Pipelines using deep learning have

[5]We define true inliers as 2D keypoint detections such that there exists a ground-truth annotated keypoint with the same ID within a radius of 5 pixels.

seen great successes in areas such as human pose estimation [105], [66], [103], [36], [62], and pose estimation of household objects [60], [32], [73]. With the growing interest in self-driving vehicles, research has also focused on jointly estimating vehicle pose and shape [15], [41], [57], [46], [97].

For methods that aim to recover both the pose and shape of objects, a common paradigm is to use **end-to-end methods**. Usually, an encoder-decoder network is used to first convert input images to some latent representations, and then map the latent representation back to 3D estimates (*e.g.,* 3D bounding boxes, or pose and shape estimates). For example, Richter *et al.* [79] predict 3D shapes through an efficient 2D encoding. Groueix *et al.* [33] represent shapes as collections of parametric surface elements. Tatarchenko *et al.* [98] generate 3D shapes through an octree representation. Burchfiel *et al.* [12] train CNNs with generative representations of 3D objects to predict probabilistic distributions of object poses. An additional alignment loss can also be incorporated into the network to directly regress a pose [4], [58], [59]. Wen *et al.* [113] design a network with a loss function over $\mathrm{SE}(3)$ to regress relative poses. One drawback of such approaches is that it is difficult for neural networks to learn the necessary 3D structure of the object on a per-pixel basis; moreover, end-to-end approaches typically require 3D pose labels that might be difficult (or expensive) to obtain for real data. As shown by Tatarchenko *et al.* [99], such networks can be outperformed by methods trained on model recognition and retrieval only. Alternative methods circumvent pose and shape estimation and directly regress 3D semantic keypoints for manipulation [60] or dense visual descriptors [31].

Multi-stage methods constitute another major paradigm for category-level perception. Such approaches commonly include a neural-network-based front-end that extracts features from input data (such as RGB or RGB-D images) [73], [21], and an optimization-based back-end that recovers the 3D pose of the object given the features [73], [67], [74], [88], [38]. The front-ends may predict positions of semantic keypoints [73], or feature embeddings [21], and generate correspondences from those features. In early works, Lim *et al.* [55] establish 2D-3D correspondences between images and textureless CAD models by using HOG descriptors on images and rendered edgemaps of the CAD models. Chabot *et al.* [15] regress a set of 2D part coordinates, and then choose the best corresponding 3D template. Pavlakos *et al.* [73] use a stacked hourglass neural network [66] for 2D semantic keypoint detection. In other works, a canonical category-level coordinate space is predicted for each detection, from which correspondences are generated [109], [29], [54], [17]. Our work belongs to the class of multi-stage methods. In particular, we use [41] as our front-end, and develop optimal and robust back-end solvers.

Research effort has also been devoted to developing more robust and efficient **back-end solvers** given 2D or 3D features extracted by the front-end. The back-end solvers recover the 3D pose (and possibly the shape) of the object by solving an optimization problem [67], [73], [74], [88], [38]. Depending on the input modalities, back-end solvers can be roughly divided into 2D-3D or 3D-3D solvers, where the former use 2D inputs only, and the latter incorporate additional depth information. A number of related works investigate **2D-3D back-end solvers** [73], [102], [43], [66], [125], [126], [86]. Hou *et al.* [38] defer the task of shape estimation to a neural network, and use EPnP [52] to solve for the object bounding box's pose only. Zhou *et al.* [125], [126] propose a convex relaxation to jointly optimize 3D shape parameters and object pose from 2D keypoints. However, the relaxation assumes a weak perspective camera model, which might lead to poor results if the object is close to the camera. Yang and Carlone [117] apply the moment/sums-of-squares hierarchy [10], [49] to develop tighter relaxations than [125], still under the assumption of a weak perspective model. Schmeckpeper *et al.* [86] use a local solver with a full perspective camera model. Our work differs from [125], [126], [117], [86] since we propose a *certifiably optimal solver for the full perspective case*, using an algebraic point-to-line cost.

3D-3D back-end solvers have been investigated in the robotics literature [112], [93], [108]. In robotics applications such as manipulation and self-driving, depth information is readily available either via direct sensing (*e.g.,* RGB-D or stereo) or algorithms (*e.g.,* mono depth techniques [25], [48]), so the requirements of depth is not too constraining. Wang *et al.* [107] decouple shape from pose estimation by predicting category specific keypoints and use Arun's method [2] for estimating frame-by-frame relative pose. Wen *et al.* [112] view category-level object detection and tracking as a pose graph optimization problem, solving 3D registration of keypoints across frames and then jointly optimizing the pose graph online. Deng *et al.* [21] use nonlinear optimization, and alternate between optimizing shape size and pose. In this and our previous work [89], we propose the first 3D-3D certifiably optimal solver that runs in a fraction of a second even in the presence of thousands of CAD models.

B. Robust Estimation

We review three robust estimation paradigms: *M-estimation*, *consensus maximization*, and *graph-based outlier pruning*.

M-Estimation performs estimation by optimizing a robust cost function that reduces the influence of outliers. The resulting problems are typically optimized using iterative local solvers. MacTavish and Barfoot [100] compare several robust costs using iterative re-weighted least squares solvers. The downside of local solvers is that they need a good initial guess, which is often unavailable in practice. A popular approach to circumvent the need for an initial guess is *Graduated Non-Convexity* (GNC) [9], [8]. Zhou *et al.* [124] use GNC for point cloud registration. Yang *et al.* [115] and Antonante *et al.* [1] combine GNC with global non-minimal solvers and show their general applicability to problems with up to 80% outliers.

For certain low-dimensional geometric problems, fast global solvers exist. Enqvist *et al.* [26] use a *truncated least squares* (TLS) cost to solve triangulation problems in polynomial time in the number of measurements, but exponential time in the dimension of the to-be-estimated state $\boldsymbol{x}$. Ask *et al.* [3] use a TLS cost for image registration. Recently, *certifiably-robust* globally optimal solvers based on *convex relaxations* have been used for *M-estimation* [118], [47], [121], [116]. Carlone and Calafiore [14] and Lajoie *et al.* [47] study SDP relaxations for pose graph optimization. Yang *et al.* [121] develop an SDP relaxation for point cloud registration. Unfortunately, due to the poor scalability of current SDP solvers, such methods are mostly viable to *check* optimality [121], [118].

Consensus Maximization is a framework for robust estimation that aims to find the largest set of measurements with errors below a user-defined threshold. Consensus maximization is NP-hard [18], [1], hence the community has resorted to randomized approaches, such as RANSAC [30]. RANSAC repeatedly draws a minimal subset of measurements from which a rough estimate is computed, and the algorithm stops after finding an estimate that agrees with a large set of measurements. While RANSAC works well for problems where the minimal set is small and there are not many outliers, the average number of iterations it requires increases exponentially with the percentage of outliers [76], making it impractical for problems with many outliers. On the other hand, global solvers, such as branch-and-bound (BnB) [53] and tree search [13], exist but scale poorly with the problem size, with BnB being exponential in the size of $\boldsymbol{x}$, and tree search being exponential in the number of outliers [13].

Graph-based Outlier pruning methods aim at discarding gross outliers from the set of measurements. These methods do not necessarily reject all the outliers, hence they are often used as a preprocessing for M-estimation or maximum consensus [121], [90]. Outlier pruning methods detect outliers by analyzing a *compatibility graph*, where vertices represent data points and edges represent pre-defined compatibility measures between data points [90]. Bailey *et al.* [5] propose a Maximum Common Subgraph algorithm for feature matching in lidar scans. Segundo and Artieda [84] build an association graph and find the maximum clique for 2D image feature matching. Perera and Barnes [75] segment objects under rigid body motion with a clique formulation. Leordeanu and Hebert [51] establish image matches by finding strongly-connected clusters in the correspondence graph with an approximate spectral method. Enqvist *et al.* [27] develop an outlier rejection algorithm for 3D-3D and 2D-3D registration based on approximate vertex cover. Recent progress in graph algorithms (*e.g.,* [82], [71]) has led to fast graph-theoretic outlier pruning algorithms that are robust to extreme outlier rates, see, *e.g.,*TEASER++ [121].

In this work we generalize graph-based methods to use **hypergraphs**: while a standard graph only contains edges connecting pairs of nodes (which represent compatibility tests in our context), each edge in a hypergraph may connect an arbitrary subset of vertices. Hypergraphs have been studied in the context of network learning, robotics, and computer vision. Torres-Jimenez [104] develops an exact algorithm for finding maximum cliques in uniform hypergraphs. Shun [91] develops a collection of fast parallel hypergraph algorithms for large-scale networks. Srinivasan *et al.* [95] develop a framework for hypergraph representation learning. Rueb *et al.* [83] formulate free space as a hypergraph for robot path planning. Du *et al.* [23] represent humans as a hypergraph for visual tracking. Yu *et al.* [122] treat image classification as a hypergraph edge weight optimization problem. In our work, we build upon [90] by extending the definition of *compatibility graphs* from simple graphs to hypergraphs.

X. Conclusion

We proposed PACE2D* and PACE3D*, the first certifiably optimal solvers for the estimation of the pose and shape of 3D objects from 2D and 3D keypoint detections, respectively. While existing iterative methods get stuck in local minima corresponding to poor estimates, PACE2D* and PACE3D* leverage tight SDP relaxations to compute certifiably optimal estimates. We then design a general framework for graph-theoretic outlier pruning, named ROBIN, that extends our original proposal in [90] to operate on compatibility hypergraphs. We show that ROBIN can be effectively applied to 2D and 3D category-level perception and is able to prune a large fraction of outliers. The combination of ROBIN and our optimal solvers (PACE2D* and PACE3D*), leads to PACE2D# and PACE3D#, which are outlier-robust algorithms for 3D-3D and 2D-3D pose and shape estimation. While PACE2D# is currently slow and is sensitive to the quality of the keypoint detections, PACE3D# largely outperforms the state of the art and a non-optimized implementation runs in a fraction of a second.

References

[1] P. Antonante, V. Tzoumas, H. Yang, and L. Carlone. Outlier-robust estimation: Hardness, minimally tuned algorithms, and applications. *IEEE Trans. Robotics*, 38(1):281–301, 2021. (pdf)

[2] K. Arun, T. Huang, and S. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Trans. Pattern Anal. Machine Intell.*, 9(5):698–700, sept. 1987.

[3] E. Ask, O. Enqvist, and F. Kahl. Optimal geometric fitting under the truncated l2-norm. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1722–1729, 2013.

[4] A. Avetisyan, A. Dai, and M. Nießner. End-to-End CAD Model Retrieval and 9DoF Alignment in 3D Scans. In *Intl. Conf. on Computer Vision (ICCV)*, 2019.

[5] T. Bailey, E. M. Nebot, J. K. Rosenblatt, and H. F. Durrant-Whyte. Data association for mobile robot navigation: a graph theoretic approach. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 3, pp. 2512–2517, 2000.

[6] A. Bandeira. A note on probably certifiably correct algorithms. *arXiv:1509.00824*, 2015.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(4):509–522, 2002.

[8] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Intl. J. of Computer Vision*, 19(1):57–91, 1996.

[9] A. Blake and A. Zisserman. *Visual reconstruction*. MIT Press, 1987.

[10] G. Blekherman, P. A. Parrilo, and R. R. Thomas. *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012.

[11] J. Briales and J. Gonzalez-Jimenez. Fast global optimality verification in 3D SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4630–4636, Oct 2016.

[12] B. Burchfiel and G. Konidaris. Probabilistic category-level pose estimation via segmentation and predicted-shape priors. *arXiv preprint arXiv:1905.12079*, 2019.

[13] Z. Cai, T.-J. Chin, and V. Koltun. Consensus maximization tree search revisited. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 1637–1645, 2019.

[14] L. Carlone and G. Calafiore. Convex relaxations for pose graph optimization with outliers. *IEEE Robotics and Automation Letters (RA-L)*, 3(2):1160–1167, 2018. arxiv preprint: 1801.02112, (pdf)

[15] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2040–2049, 2017.

[16] T. Chan and L. Vese. An active contour model without edges. In *Intl. Conf. Scale-Space Theor. Comput. Vision*, pp. 141–151. Springer, 1999.

[17] D. Chen, J. Li, Z. Wang, and K. Xu. Learning canonical shape space for category-level 6d object pose and size estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 11973–11982, 2020.

[18] T.-J. Chin, Z. Cai, and F. Neumann. Robust fitting in computer vision: Easy or hard? In *European Conf. on Computer Vision (ECCV)*, 2018.

[19] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models - their training and application. *Comput. Vis. Image Underst.*, 61(1):38–59, January 1995.

[20] E. De Klerk. *Aspects of semidefinite programming: interior point algorithms and selected applications*, vol. 65. Springer Science & Business Media, 2006.

[21] X. Deng, J. Geng, T. Bretl, Y. Xiang, and D. Fox. iCaps: iterative category-level object pose and shape estimation. *IEEE Robotics and Automation Letters*, 2022.

[22] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[23] D. Du, H. Qi, L. Wen, Q. Tian, Q. Huang, and S. Lyu. Geometric hypergraph learning for visual tracking. *IEEE Trans. Cybern.*, 47(12):4182–4195, 2016.

[24] D. Eberly. *3D game engine design: a practical approach to real-time computer graphics*. CRC Press, 2006.

[25] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2366–2374, 2014.

[26] O. Enqvist, E. Ask, F. Kahl, and K. Åström. Robust fitting for multiple view geometry. In *European Conf. on Computer Vision (ECCV)*, pp. 738–751. Springer, 2012.

[27] O. Enqvist, K. Josephson, and F. Kahl. Optimal correspondences from pairwise constraints. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 1295–1302, 2009.

[28] A. Eriksson, C. Olsson, F. Kahl, and T.-J. Chin. Rotation averaging and strong duality. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[29] Q. Feng and N. Atanasov. Fully convolutional geometric features for category-level object alignment. *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 8492–8498, 2020.

[30] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. ACM*, 24:381–395, 1981.

[31] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

[32] W. Gao and R. Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE Robotics and Automation Letter (RA-L)*, 2020.

[33] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. A papier-mâché approach to learning 3d surface generation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 216–224, 2018.

[34] L. Gu and T. Kanade. 3D alignment of face in a single image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 1305–1312, 2006.

[35] R. Hartley and F. Kahl. Global optimization through rotation space search. *Intl. J. of Computer Vision*, 82(1):64–79, 2009.

[36] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[37] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Amer.*, 4(4):629–642, Apr 1987.

[38] T. Hou, A. Ahmadyan, L. Zhang, J. Wei, and M. Grundmann. Mobilepose: Real-time pose estimation for unseen objects with weak shape supervision. *arXiv preprint arXiv:2003.03522*, 2020.

[39] J. F. Hughes, A. Van Dam, M. McGuire, J. D. Foley, D. Sklar, S. K. Feiner, and K. Akeley. *Computer graphics: principles and practice*. Pearson Education, 2014.

[40] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *Intl. J. of Computer Vision*, 1(4):321–331, 1987.

[41] L. Ke, S. Li, Y. Sun, Y.-W. Tai, and C.-K. Tang. GSNet: joint vehicle pose and shape reconstruction with geometrical and scene-aware supervision. In *European Conf. on Computer Vision (ECCV)*, pp. 515–532. Springer, 2020.

[42] L. Kneip, H. Li, and Y. Seo. UPnP: An optimal o(n) solution to the absolute pose problem with universal applicability. In *European Conf. on Computer Vision (ECCV)*, pp. 127–142. Springer, 2014.

[43] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis. Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 2252–2261, 2019.

[44] N. Kolotouros, G. Pavlakos, and K. Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.

[46] A. Kundu, Y. Li, and J. M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3559–3568, 2018.

[47] P. Lajoie, S. Hu, G. Beltrame, and L. Carlone. Modeling perceptual aliasing in SLAM via discrete-continuous graphical models. *IEEE Robotics and Automation Letters (RA-L)*, 2019. extended ArXiv version: (pdf). Supplemental Material: (pdf).

[48] K. Lasinger, R. Ranftl, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019.

[49] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optim.*, 11(3):796–817, 2001.

[50] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[51] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Intl. Conf. on Computer Vision (ICCV)*, vol. 2, pp. 1482–1489. IEEE, 2005.

[52] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *Intl. J. of Computer Vision*, 81(2):155, 2009.

[53] H. Li. Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 1074–1080, 2009.

[54] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song. Category-level articulated object pose estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3706–3715, 2020.

[55] J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA objects: Fine pose estimation. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 2992–2999, 2013.

[56] Y.-L. Lin, V. I. Morariu, W. H. Hsu, and L. S. Davis. Jointly optimizing 3D model fitting and fine-grained classification. In *European Conf. on Computer Vision (ECCV)*, 2014.

[57] J. G. López, A. Agudo, and F. Moreno-Noguer. Vehicle pose estimation via regression of semantic points of interest. In *Intl. Symp. on Image and Signal Processing and Analysis (ISPA)*, pp. 209–214. IEEE, 2019.

[58] F. Manhardt, W. Kehl, and A. Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2069–2078, 2019.

[59] F. Manhardt, M. Nickel, S. Meier, L. Minciullo, and N. Navab. Cps++: Class-level 6d pose and shape estimation from monocular images. *arXiv preprint arXiv:2003.05848*, 2020.

[60] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, 2019.

[61] F. L. Markley. Attitude determination using vector observations and the singular value decomposition. *J. of the Astronautical Sciences*, 36(3):245–258, 1988.

[62] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose estimation. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 2640–2649, 2017.

[63] P. McCausland. Self-driving Uber car that hit and killed woman did not recognize that pedestrians jaywalk. *NBC News*, Nov 2019.

[64] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017.

[65] J. Mundy and A. Zisserman. *Geometric invariance in computer vision*. MIT Press, Cambridge, MA, USA, 1992.

[66] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conf. on Computer Vision (ECCV)*, pp. 483–499. Springer, 2016.

[67] M. Oberweger, M. Rad, and V. Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *European Conf. on Computer Vision (ECCV)*, pp. 119–134, 2018.

[68] C. Olsson, F. Kahl, and M. Oskarsson. Optimal estimation of perspective camera pose. In *Intl. Conf. on Pattern Recognition (ICPR)*, vol. 2, pp. 5–8. IEEE, 2006.

[69] M. Pantic and L. J. M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(12):1424–1445, 2000.

[70] A. Parra Bustos and T. J. Chin. Guaranteed outlier removal for point cloud registration with correspondences. *IEEE Trans. Pattern Anal. Machine Intell.*, 40(12):2868–2882, 2018.

[71] A. Parra Bustos, T.-J. Chin, F. Neumann, T. Friedrich, and M. Katzmann. A practical maximum clique algorithm for matching with pairwise constraints. *arXiv preprint arXiv:1902.01534*, 2019.

[72] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320, 2003.

[73] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017.

[74] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 4561–4570, 2019.

[75] S. Perera and N. Barnes. Maximal cliques based rigid body motion segmentation with a RGB-D camera. In *Asian Conf. on Computer Vision*, pp. 120–133. Springer, 2012.

[76] R. Raguram, J.-M. Frahm, and M. Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *European Conf. on Computer Vision (ECCV)*, pp. 500–513. Springer, 2008.

[77] V. Ramakrishna, T. Kanade, and Y. Sheikh. Reconstructing 3D human

pose from 2D image landmarks. In *European Conf. on Computer Vision (ECCV)*, 2012.

[78] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[79] S. R. Richter and S. Roth. Matryoshka networks: Predicting 3D geometry via nested shape layers. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1936–1944, 2018.

[80] R. T. Rockafellar. *Convex analysis*, vol. 36. Princeton Univ. Press, 1970.

[81] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard. SE-Sync: a certifiably correct algorithm for synchronization over the Special Euclidean group. *Intl. J. of Robotics Research*, 2018. arxiv preprint: 1611.00128, (pdf).

[82] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin. Parallel maximum clique algorithms with applications to network analysis. *SIAM J. on Scientific Computing*, 37(5):C589–C616, 2015.

[83] K. D. Rueb and A. K. Wong. Structuring free space as a hypergraph for roving robot path planning and navigation. *IEEE Trans. Pattern Anal. Machine Intell.*, (2):263–273, 1987.

[84] P. San Segundo and J. Artieda. A novel clique formulation for the visual feature matching problem. *Appl. Intelligence*, 43(2):325–342, 2015.

[85] S. Schaible and J. Shi. Fractional programming: the sum-of-ratios case. *Optimization Methods and Software*, 18(2):219–229, 2003.

[86] K. Schmeckpeper, P. R. Osteen, Y. Wang, G. Pavlakos, K. Chaney, W. Jordan, X. Zhou, K. G. Derpanis, and K. Daniilidis. Semantic keypoint-based pose estimation from single rgb frames. *arXiv preprint arXiv:2204.05864*, 2022.

[87] G. Schweighofer and A. Pinz. Globally optimal O(n) solution to the PnP problem for general camera models. In *British Machine Vision Conf. (BMVC)*, pp. 1–10, 2008.

[88] M. Shan, Q. Feng, and N. Atanasov. Object residual constrained visual-inertial odometry. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5104–5111, 2020.

[89] J. Shi, H. Yang, and L. Carlone. Optimal pose and shape estimation for category-level 3D object perception. In *Robotics: Science and Systems (RSS)*, 2021. arXiv preprint arXiv: 2104.08383, (pdf), (video).

[90] J. Shi, H. Yang, and L. Carlone. ROBIN: a graph-theoretic approach to reject outliers in robust estimation using invariants. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021. arXiv preprint arXiv: 2011.03659, (pdf).

[91] J. Shun. Practical parallel hypergraph algorithms. In *ACM SIGPLAN Symp. on Princ. and Pract. of Parallel Program.*, pp. 232–249, 2020.

[92] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Intl. Conf. on Learning Representations*, 2015.

[93] M. Slavcheva, W. Kehl, N. Navab, and S. Ilic. Sdf-2-sdf: Highly accurate 3d object reconstruction. In *European Conf. on Computer Vision (ECCV)*, pp. 680–696. Springer, 2016.

[94] X. Song, P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su, H. Li, and R. Yang. ApolloCar3D: A large 3d car instance understanding benchmark for autonomous driving. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5452–5462, 2019.

[95] B. Srinivasan, D. Zheng, and G. Karypis. Learning over families of sets-hypergraph representation learning for higher order tasks. In *SIAM Intl. Conf. on Data Mining*, pp. 756–764. SIAM, 2021.

[96] G. Strang. *Introduction to linear algebra*. Cambridge Press, Wellesley, MA, 5th edition, 2016.

[97] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *arXiv preprint arXiv:1807.03146*, 2018.

[98] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 2088–2096, 2017.

[99] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox. What Do Single-view 3D Reconstruction Networks Learn? In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3405–3414, 2019.

[100] K. M. Tavish and T. D. Barfoot. At all costs: A comparison of robust cost functions for camera correspondence outliers. In *Conf. Computer and Robot Vision*, pp. 62–69. IEEE, 2015.

[101] A. N. Tikhonov, A. Goncharsky, V. Stepanov, and A. G. Yagola. *Numerical methods for the solution of ill-posed problems*, vol. 328. Springer Science & Business Media, 2013.

[102] D. Tome, C. Russell, and L. Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2500–2509, 2017.

[103] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *arXiv preprint arXiv:1406.2984*, 2014.

[104] J. Torres-Jimenez, J. C. Perez-Torres, and G. Maldonado-Martinez. hclique: An exact algorithm for maximum clique problem in uniform hypergraphs. *Discrete Math., Algo. and Appl.*, 9(06):1750078, 2017.

[105] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1653–1660, 2014.

[106] R. Tron, D. Rosen, and L. Carlone. On the inclusion of determinant constraints in lagrangian duality for 3D SLAM. In *Robotics: Science and Systems (RSS), Workshop "The problem of mobile sensors: Setting future goals and indicators of progress for SLAM"*, 2015. (pdf).

[107] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 10059–10066. IEEE, 2020.

[108] F. Wang and K. Hauser. In-hand object scanning via rgb-d video segmentation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3296–3302. IEEE, 2019.

[109] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2642–2651, 2019.

[110] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang. The ApolloScape open dataset for autonomous driving and its application. *IEEE Trans. Pattern Anal. Machine Intell.*, 2019.

[111] W. Wang and M. A. Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.

[112] B. Wen and K. Bekris. Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 8067–8074. IEEE, 2021.

[113] B. Wen, C. Mitash, B. Ren, and K. E. Bekris. se(3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 10367–10373. IEEE, 2020.

[114] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond PASCAL: A benchmark for 3d object detection in the wild. In *IEEE Winter Conf. on Appl. of Computer Vision*, pp. 75–82. IEEE, 2014.

[115] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):1127–1134, 2020. arXiv preprint arXiv:1909.08605 (with supplemental material), (pdf).

[116] H. Yang and L. Carlone. A quaternion-based certifiably optimal solution to the Wahba problem with outliers. In *Intl. Conf. on Computer Vision (ICCV)*, 2019. (Oral Presentation, accept rate: 4%), Arxiv version: 1905.12536, (pdf).

[117] H. Yang and L. Carlone. In perfect shape: Certifiably optimal 3D shape reconstruction from 2D landmarks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. Arxiv version: 1911.11924, (pdf).

[118] H. Yang and L. Carlone. One ring to rule them all: Certifiably robust geometric perception with outliers. In *Conf. on Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 18846–18859, 2020. (pdf).

[119] H. Yang and L. Carlone. Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization. *IEEE Trans. Pattern Anal. Machine Intell.*, 2021. (pdf).

[120] H. Yang, C. Doran, and J.-J. Slotine. Dynamical pose estimation. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 5926–5935, 2021.

[121] H. Yang, J. Shi, and L. Carlone. TEASER: Fast and Certifiable Point Cloud Registration. *IEEE Trans. Robotics*, 37(2):314–333, 2020. extended arXiv version 2001.07715 (pdf).

[122] J. Yu, D. Tao, and M. Wang. Adaptive hypergraph learning and its application in image classification. *IEEE Trans. Image Processing*, 21(7):3262–3272, 2012.

[123] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi. Revisiting the PnP problem: A fast, general and optimal solution. In *Intl. Conf. on Computer Vision (ICCV)*, pp. 2344–2351, 2013.

[124] Q. Zhou, J. Park, and V. Koltun. Fast global registration. In *European Conf. on Computer Vision (ECCV)*, pp. 766–782. Springer, 2016.

[125] X. Zhou, S. Leonardos, X. Hu, and K. Daniilidis. 3D shape reconstruction from 2D landmarks: A convex formulation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[126] X. Zhou, M. Zhu, S. Leonardos, and K. Daniilidis. Sparse representation for 3D shape estimation: A convex relaxation approach. *IEEE Trans. Pattern Anal. Machine Intell.*, 39(8):1648–1661, 2017.

APPENDIX A
PROOF OF THEOREM 7: INLIERS BELONG TO A CLIQUE

Proof: By definition, compatibility tests are designed to pass as long as the subset of n nodes under test includes all inliers. Therefore, the set of nodes corresponding to inliers, say $\mathcal{I}$, will be such that any subset of n nodes in $\mathcal{I}$ will be connected by a hyperedge, and therefore will form a hyperclique in the compatibility hypergraph. ■

APPENDIX B
COMPARISON BETWEEN CLIQUE-EXPANDED GRAPHS AND HYPERGRAPHS FOR ROBIN

Comparison with [90]. In our previous work [90] — where we first proposed ROBIN— we defined compatibility graphs as ordinary graphs, and inliers structures as maximum cliques. In the present paper, we define compatibility graphs as hypergraphs (Definition 5), and inlier structures as maximum hypercliques (Theorem 7). This new formulation is equivalent to [90] for 2-invariants, but different otherwise. Topologically, the compatibility graphs in [90] can be seen as *clique-expanded* compatibility hypergraphs, where each hyperedge on a subset of n nodes is substituted with pairwise edges between all nodes in the subset (*i.e.,* a clique in the graph). Compared to [90], our new formulation leads to pruning a larger number of outliers, as shown in the example below.

Example. Consider a 3-invariant and its compatibility test, with 5 measurements, denoted as Nodes 1 to 5. Node 5 is an outlier, whereas Node 1 through 4 are inliers. Assume Table A3 contains the results of running the compatibility test with the 3-invariant. Note that $(1,2,5)$, $(2,3,5)$, and $(2,4,5)$ pass the compatibility test despite Node 5 being an outlier. Fig. A1 shows the compatibility hypergraph constructed according to the results, with the maximum hyperclique being $(1,2,3,4)$. Fig. A2 shows the clique-expanded hypergraph. In this case, the maximum clique is $(1,2,3,4,5)$, which is the entire measurement set including the outlier. Thus, for this example, ROBIN described in this paper will successfully reject Node 5 as an outlier, while ROBIN described in [90] will not.

Triplet	Pass the Compatibility Test?
1,2,3	True
1,2,4	True
1,2,5	True
1,3,4	True
1,3,5	False
1,4,5	False
2,3,4	True
2,3,5	True
2,4,5	True
3,4,5	False

TABLE A3
RESULTS OF RUNNING THE COMPATIBILITY TEST ON ALL POSSIBLE TRIPLETS.

While this is only a toy example, we empirically observe that using hypergraphs improves the effectiveness of outlier pruning in real problems, such as the ones in Section V-B.

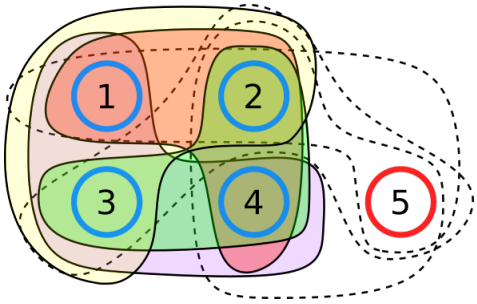


Fig. A1. Compatibility hypergraph with hyperedges including the outlier (outlier is Node 5 circled in red; hyperedges containing the outlier are represented with dashed edges). The maximum hyperclique is $(1,2,3,4)$.

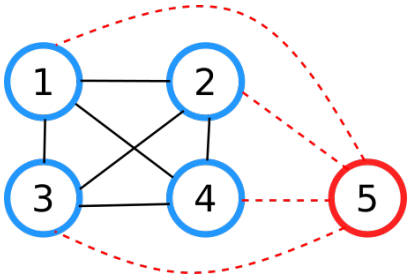


Fig. A2. Clique-expanded compatibility graph. Outlier is Node 5; all edges including the outlier are marked in red, with dashed lines. Comparing to the hypergraph where the maximum hyperclique is $(1,2,3,4)$, in this graph the maximum clique is $(1,2,3,4,5)$.

APPENDIX C
FINDING MAXIMUM HYPERCLIQUES IN COMPATIBILITY HYPERGRAPHS

While there exist fast parallel implementations for finding the maximum clique in ordinary graphs (see, *e.g.,* [82]), to the best of our knowledge there is no such implementation for hypergraphs. Thus, in this paper we use a simple mixed-integer linear programming (MILP) formulation for finding the maximum hyperclique within an n-uniform hypergraph.

Assume an arbitrary n-uniform hypergraph $\mathcal{G}(\mathcal{V},\mathcal{E})$, where $|\mathcal{V}|= N$. Consider the following MILP:

$$\begin{aligned} \max_{\boldsymbol{b}\in\{0,1\}^N} \quad & \textstyle\sum_{i=1}^{N} b_i \\ \text{s.t.} \quad & \textstyle\sum_{i\in\mathcal{M}} b_i \le n-1, \quad \forall \mathcal{M}\subset\mathcal{V}, |\mathcal{M}|= n, \mathcal{M}\notin\mathcal{E} \end{aligned} \tag{MILP}$$

where $\mathcal{E}$ is the set of hyperedges, $\mathcal{M}\subset\mathcal{V}$ are subsets of n nodes, and b_i, $i=1,\dots,N$, are binary variables that indicate whether node i belongs to the maximum hyperclique.

Intuitively, (MILP) looks for a set of nodes such that — within that set— there is no subset of n nodes that are not connected by a hyperedge. This matches the definition of a hyperclique, as formalized below.

Theorem A1 (MILP Finds Maximum Hypercliques). *For any feasible solution $\boldsymbol{b}$ to (MILP), the set of nodes $\{i\in\mathcal{V}: b_i=1\}$ forms a hyperclique in $\mathcal{G}$; moreover, the optimal solution $b^\star$ is such that the set $\{i\in\mathcal{V}: b_i^\star=1\}$ is a* maximum *hyperclique.*

Proof: Assume by contradiction that there exists a feasible solution of the MILP that does *not* form a hyperclique.

Without loss of generality, assume that —for some m— such a solution includes nodes $\hat{\mathcal{V}} \doteq \{v_1, v_2, \ldots, v_m\}$, *i.e.,* $b_i = 1$ for $i = 1, \ldots, m$, and zero otherwise. Since this set of nodes does not form a hyperclique, there is a subset $\mathcal{M}$ of n nodes in $\hat{\mathcal{V}}$ that does not belong to the hyperedge set $\mathcal{E}$, and for such subset $\sum_{i \in \mathcal{M}} b_i = n$. which violates the constraint in the MILP. Since $\hat{\mathcal{V}}$ violates a constraint, it cannot be feasible, leading to a contradiction. Since the objective maximizes the count of non-zero b_i (*i.e.,* the number of nodes in the hyperclique), the optimal solution is a maximum hyperclique. ∎

Appendix D
Minimum and Maximum Distances between Convex Hulls for Eq. (17)

Recall from eq. (17) the definitions of $b_{ij}^{\min}$ and $b_{ij}^{\max}$:

$$b_{ij}^{\min} = \min_{\boldsymbol{c} \geq 0, \mathbf{1}^\mathsf{T}\boldsymbol{c}=1} \left\| \sum_{k=1}^{K} c_k (\boldsymbol{b}_j^k - \boldsymbol{b}_i^k) \right\|, \tag{A1}$$

$$b_{ij}^{\max} = \max_{\boldsymbol{c} \geq 0, \mathbf{1}^\mathsf{T}\boldsymbol{c}=1} \left\| \sum_{k=1}^{K} c_k (\boldsymbol{b}_j^k - \boldsymbol{b}_i^k) \right\|, \tag{A2}$$

and let us use the following shorthand:

$$\boldsymbol{b}_{ij}^k \triangleq \boldsymbol{b}_j^k - \boldsymbol{b}_i^k, \tag{A3}$$

$$\boldsymbol{B}_{ij} \triangleq \begin{bmatrix} \boldsymbol{b}_{ij}^1 & \cdots & \boldsymbol{b}_{ij}^K \end{bmatrix} \in \mathbb{R}^{3 \times K}, \tag{A4}$$

to write problems (A1) and (A2) compactly as:

$$b_{ij}^{\min} = \min_{\boldsymbol{c} \geq 0, \mathbf{1}^\mathsf{T}\boldsymbol{c}=1} \|\boldsymbol{B}_{ij}\boldsymbol{c}\|, \quad b_{ij}^{\max} = \max_{\boldsymbol{c} \geq 0, \mathbf{1}^\mathsf{T}\boldsymbol{c}=1} \|\boldsymbol{B}_{ij}\boldsymbol{c}\|. \tag{A5}$$

Compute $b_{ij}^{\max}$. Because $\|\boldsymbol{B}_{ij}\boldsymbol{c}\|$ is a convex function of $\boldsymbol{c}$, and the maximum of a convex function over a polyhedral set (in our case, the standard simplex $\Delta_K \triangleq \{\boldsymbol{c} \in \mathbb{R}^K : \boldsymbol{c} \geq 0, \mathbf{1}^\mathsf{T}\boldsymbol{c} = 1\}$) is always obtained at one of the vertices of the polyhedron [80, Corollary 32.3.4], we have:

$$b_{ij}^{\max} = \max_k \left\|\boldsymbol{b}_{ij}^k\right\|, \tag{A6}$$

since the vertices of Δ_K are the vectors $\boldsymbol{e}_k, k = 1, \ldots, K$, where $\boldsymbol{e}_k$ is one at its k-th entry and zero anywhere else.

Compute $b_{ij}^{\min}$. Observe that computing the minimum of $\|\boldsymbol{B}_{ij}\boldsymbol{c}\|$ is equivalent to computing the minimum of $\|\boldsymbol{B}_{ij}\boldsymbol{c}\|^2 = \boldsymbol{c}^\mathsf{T}(\boldsymbol{B}_{ij}^\mathsf{T}\boldsymbol{B}_{ij})\boldsymbol{c}$ because the quadratic function $f(x) = x^2$ is monotonically increasing in the interval $[0, \infty]$, and hence we first solve the following convex quadratic program (QP):

$$\min_{\boldsymbol{c} \in \mathbb{R}^K} \quad \boldsymbol{c}^\mathsf{T}(\boldsymbol{B}_{ij}^\mathsf{T}\boldsymbol{B}_{ij})\boldsymbol{c} \tag{A7}$$

$$\text{s.t.} \quad \boldsymbol{c} \geq 0, \quad \mathbf{1}^\mathsf{T}\boldsymbol{c} = 1 \tag{A8}$$

and then compute $b_{ij}^{\min} = \|\boldsymbol{B}_{ij}\boldsymbol{c}^\star\|$ from the solution $\boldsymbol{c}^\star$ of the QP. Note that the QP (A7) can be solved in milliseconds for large K, so pre-computing $b_{ij}^{\min}$ for all $1 \leq i < j \leq N$ is still tractable even when N is large.

Appendix E
Proof of Proposition 10: 2D Winding Orders

Proof: Consider the 2D image points $\boldsymbol{p}_{2\text{D},i}$, $\boldsymbol{p}_{2\text{D},j}$, and $\boldsymbol{p}_{2\text{D},m}$ and their representation in homogeneous coordinates:

$$\bar{\boldsymbol{p}}_{2\text{D},i} = \begin{bmatrix} \boldsymbol{p}_{2\text{D},i} \\ 1 \end{bmatrix} \quad \bar{\boldsymbol{p}}_{2\text{D},j} = \begin{bmatrix} \boldsymbol{p}_{2\text{D},j} \\ 1 \end{bmatrix} \quad \bar{\boldsymbol{p}}_{2\text{D},m} = \begin{bmatrix} \boldsymbol{p}_{2\text{D},m} \\ 1 \end{bmatrix}.$$

Recall these 3D vectors are expressed in a standard right-handed image coordinate frame with origin at the center of the image (irrelevant for the derivation below), and where the $\boldsymbol{x}$ axis points towards the right in the image plane, the $\boldsymbol{y}$ axis points down in the image plane, and the $\boldsymbol{z}$ axis points into the image plane (to ensure right-handedness).

Now the triplet of points is arranged in clockwise order if their crossproduct is aligned with the $\boldsymbol{z}$ axis:

$$(\bar{\boldsymbol{p}}_{2\text{D},j} - \bar{\boldsymbol{p}}_{2\text{D},i}) \times (\bar{\boldsymbol{p}}_{2\text{D},m} - \bar{\boldsymbol{p}}_{2\text{D},i}) \tag{A9}$$

$$= \det \left(\begin{bmatrix} \boldsymbol{x} & \boldsymbol{y} & \boldsymbol{z} \\ (\boldsymbol{p}_{2\text{D},j} - \boldsymbol{p}_{2\text{D},i})^\mathsf{T} & 0 \\ (\boldsymbol{p}_{2\text{D},m} - \boldsymbol{p}_{2\text{D},i})^\mathsf{T} & 0 \end{bmatrix} \right) \tag{A10}$$

$$= \det \left(\begin{bmatrix} (\boldsymbol{p}_{2\text{D},j} - \boldsymbol{p}_{2\text{D},i})^\mathsf{T} \\ (\boldsymbol{p}_{2\text{D},m} - \boldsymbol{p}_{2\text{D},i})^\mathsf{T} \end{bmatrix} \right) \boldsymbol{z} \tag{A11}$$

$$= \det \left(\begin{bmatrix} \boldsymbol{p}_{2\text{D},j} - \boldsymbol{p}_{2\text{D},i} & \boldsymbol{p}_{2\text{D},m} - \boldsymbol{p}_{2\text{D},i} \end{bmatrix} \right) \boldsymbol{z} \tag{A12}$$

where we used the standard relation between the cross product and the determinant and then developed the expression of the determinant. Therefore, we have

$$\det \left(\begin{bmatrix} \boldsymbol{p}_{2\text{D},j} - \boldsymbol{p}_{2\text{D},i} & \boldsymbol{p}_{2\text{D},m} - \boldsymbol{p}_{2\text{D},i} \end{bmatrix} \right) > 0 \tag{A13}$$

if the points are arranged clockwise. And

$$\det \left(\begin{bmatrix} \boldsymbol{p}_{2\text{D},j} - \boldsymbol{p}_{2\text{D},i} & \boldsymbol{p}_{2\text{D},m} - \boldsymbol{p}_{2\text{D},i} \end{bmatrix} \right) < 0 \tag{A14}$$

if the points are arranged counter-clockwise. ∎

Appendix F
Proof for Theorem 11: 3D Winding Orders

We first recall a simple fact about the scalar triple product.

Lemma A2 (Scalar Triple Product As Determinant). *For arbitrary* $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c} \in \mathbb{R}^3$,

$$\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = \det \begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} & \boldsymbol{c} \end{bmatrix} \tag{A15}$$

This is a well-known property and can be proven by inspection. We are now ready to prove Theorem 11.

Proof: Recall that $\boldsymbol{o}$ is the optical center of the camera in the CAD model's frame. Since $(\boldsymbol{R}, \boldsymbol{t})$ is the pose of the object in the coordinate frame of the camera, it follows that $\boldsymbol{o} = -\boldsymbol{R}^\mathsf{T}\boldsymbol{t}$.

For an arbitrary camera center $\boldsymbol{o}$, it follows that

$$
\begin{aligned}
&(\boldsymbol{o}-\boldsymbol{b}_i^k)\cdot\boldsymbol{n}_{i,j,m}^k \\
&\text{(using the definition of } \boldsymbol{n}_{i,j,m}^k \text{ in (20) and Lemma A2)} \\
&=\det\left(\begin{bmatrix}\boldsymbol{o}-\boldsymbol{b}_i^k & \boldsymbol{b}_j^k-\boldsymbol{b}_i^k & \boldsymbol{b}_m^k-\boldsymbol{b}_i^k\end{bmatrix}\right) && \text{(A16)} \\
&\text{(subtracting the first column from the second and third)} \\
&=\det\left(\begin{bmatrix}\boldsymbol{o}-\boldsymbol{b}_i^k & \boldsymbol{b}_j^k-\boldsymbol{o} & \boldsymbol{b}_m^k-\boldsymbol{o}\end{bmatrix}\right) && \text{(A17)} \\
&\text{(flipping the sign of the first column)} \\
&=-\det\left(\begin{bmatrix}\boldsymbol{b}_i^k-\boldsymbol{o} & \boldsymbol{b}_j^k-\boldsymbol{o} & \boldsymbol{b}_m^k-\boldsymbol{o}\end{bmatrix}\right) && \text{(A18)} \\
&\text{(multiplying the matrix by a rotation matrix } \boldsymbol{R}) \\
&=-\det\left(\begin{bmatrix}\boldsymbol{R}(\boldsymbol{b}_i^k-\boldsymbol{o}) & \boldsymbol{R}(\boldsymbol{b}_j^k-\boldsymbol{o}) & \boldsymbol{R}(\boldsymbol{b}_m^k-\boldsymbol{o})\end{bmatrix}\right) && \text{(A19)} \\
&\text{(using } \boldsymbol{o}=-\boldsymbol{R}^\top\boldsymbol{t} \text{ and } \boldsymbol{R}\boldsymbol{R}^\top=\boldsymbol{I}) \\
&=-\det\left(\begin{bmatrix}\boldsymbol{R}\boldsymbol{b}_i^k+\boldsymbol{t} & \boldsymbol{R}\boldsymbol{b}_j^k+\boldsymbol{t} & \boldsymbol{R}\boldsymbol{b}_m^k+\boldsymbol{t}\end{bmatrix}\right) && \text{(A20)} \\
&\text{(dividing each column by its third coordinate)} \\
&=-(\boldsymbol{p}_{3\mathrm{D},i})_z(\boldsymbol{p}_{3\mathrm{D},j})_z(\boldsymbol{p}_{3\mathrm{D},m})_z \\
&\quad\det\left(\begin{bmatrix}(\boldsymbol{p}_{3\mathrm{D},i})_x/(\boldsymbol{p}_{3\mathrm{D},i})_z & (\boldsymbol{p}_{3\mathrm{D},j})_x/(\boldsymbol{p}_{3\mathrm{D},j})_z & (\boldsymbol{p}_{3\mathrm{D},m})_x/(\boldsymbol{p}_{3\mathrm{D},m})_z \\ (\boldsymbol{p}_{3\mathrm{D},i})_y/(\boldsymbol{p}_{3\mathrm{D},i})_z & (\boldsymbol{p}_{3\mathrm{D},j})_y/(\boldsymbol{p}_{3\mathrm{D},j})_z & (\boldsymbol{p}_{3\mathrm{D},m})_y/(\boldsymbol{p}_{3\mathrm{D},m})_z \\ 1 & 1 & 1\end{bmatrix}\right) && \text{(A21)}
\end{aligned}
$$

In (A17) we used the fact that adding a scalar multiple of one column to another column does not change the value of the determinant, while in (A18) we observed that flipping the sign of the first column flips the sign of the determinant [96, pp. 249-252]. In (A19) we observed that left multiplying by $\boldsymbol{R}$ does not change the determinant because $\det(\boldsymbol{R})=1$. Finally, in (A21) we divided each column by its third coordinate and multiplied the determinant by the same coordinate to keep it constant [96, pp. 249-252].

Now recall that the canonical perspective projection of 3D points $\boldsymbol{p}_{3\mathrm{D},i}$, $\boldsymbol{p}_{3\mathrm{D},j}$, $\boldsymbol{p}_{3\mathrm{D},m}$ is

$$
\boldsymbol{p}_{2\mathrm{D},i}=\begin{bmatrix}(\boldsymbol{p}_{3\mathrm{D},i})_x/(\boldsymbol{p}_{3\mathrm{D},i})_z\\(\boldsymbol{p}_{3\mathrm{D},i})_y/(\boldsymbol{p}_{3\mathrm{D},i})_z\end{bmatrix}\quad \boldsymbol{p}_{2\mathrm{D},j}=\begin{bmatrix}(\boldsymbol{p}_{3\mathrm{D},j})_x/(\boldsymbol{p}_{3\mathrm{D},j})_z\\(\boldsymbol{p}_{3\mathrm{D},j})_y/(\boldsymbol{p}_{3\mathrm{D},j})_z\end{bmatrix}
$$
$$
\boldsymbol{p}_{2\mathrm{D},m}=\begin{bmatrix}(\boldsymbol{p}_{3\mathrm{D},m})_x/(\boldsymbol{p}_{3\mathrm{D},m})_z\\(\boldsymbol{p}_{3\mathrm{D},m})_y/(\boldsymbol{p}_{3\mathrm{D},m})_z\end{bmatrix}
$$

Substituting the projections back into (A21):

$$
\begin{aligned}
\text{(A21)} &= -(\boldsymbol{p}_{3\mathrm{D},i})_z(\boldsymbol{p}_{3\mathrm{D},j})_z(\boldsymbol{p}_{3\mathrm{D},m})_z \\
&\quad\det\begin{bmatrix}\boldsymbol{p}_{2\mathrm{D},i} & \boldsymbol{p}_{2\mathrm{D},j} & \boldsymbol{p}_{2\mathrm{D},m}\\ 1 & 1 & 1\end{bmatrix} \\
&= -(\boldsymbol{p}_{3\mathrm{D},i})_z(\boldsymbol{p}_{3\mathrm{D},j})_z(\boldsymbol{p}_{3\mathrm{D},m})_z \\
&\quad\det\begin{bmatrix}\boldsymbol{p}_{2\mathrm{D},i} & \boldsymbol{p}_{2\mathrm{D},j}-\boldsymbol{p}_{2\mathrm{D},i} & \boldsymbol{p}_{2\mathrm{D},m}-\boldsymbol{p}_{2\mathrm{D},i}\\ 1 & 0 & 0\end{bmatrix} && \text{(A22)}
\end{aligned}
$$

because subtracting a column from another does not change the determinant. By cofactor expansion along the last row:

$$
\begin{aligned}
\text{(A22)} &= -(\boldsymbol{p}_{3\mathrm{D},i})_z(\boldsymbol{p}_{3\mathrm{D},j})_z(\boldsymbol{p}_{3\mathrm{D},m})_z \\
&\quad\det\begin{bmatrix}\boldsymbol{p}_{2\mathrm{D},j}-\boldsymbol{p}_{2\mathrm{D},i} & \boldsymbol{p}_{2\mathrm{D},m}-\boldsymbol{p}_{2\mathrm{D},i}\end{bmatrix} && \text{(A23)}
\end{aligned}
$$

Note that we assume the object always stays in front of the camera (*i.e.,* in the direction of the positive z-axis of the camera frame). Applying the signum function, we finally have

$$
\begin{aligned}
&\operatorname{sgn}\left((\boldsymbol{o}-\boldsymbol{b}_i^k)\cdot\boldsymbol{n}_{i,j,m}^k\right) \\
&=-\operatorname{sgn}\left(\det\begin{bmatrix}\boldsymbol{p}_{2\mathrm{D},j}-\boldsymbol{p}_{2\mathrm{D},i} & \boldsymbol{p}_{2\mathrm{D},m}-\boldsymbol{p}_{2\mathrm{D},i}\end{bmatrix}\right)
\end{aligned}
$$

which proves the claim. ∎

One way to interpret Theorem 11 is that for all $\boldsymbol{o}$ in the positive half-space (resp. negative half-space) of the triplet plane, the observed winding order of the projected points is counter-clockwise (resp. clockwise) following Proposition 10.

Appendix G
Extensions of Proposition 16

This appendix discusses conditions under which Proposition 16 can be extended to the case where the 2D keypoints are generated from a convex combination of shapes —as in our generative model (3)— rather than being generated by a single shape (as currently assumed in Proposition 16).

Consider K shapes, with 2D measurements $\boldsymbol{p}_{2\mathrm{D},i}$, $i=1,\dots,N$, generated according to eq. (3), and such that $\|\boldsymbol{\epsilon}_{2\mathrm{D},i}\|<\beta$ and β is small enough for Theorem 11 to hold. Assume for each shape, we have obtained the feasible winding orders dictionary $\mathcal{W}_k$ a priori. Let $\boldsymbol{c}=\begin{bmatrix}c_1 c_2\dots c_K\end{bmatrix}$ be an arbitrary ground-truth shape coefficient vector.

Given an arbitrary camera center $\boldsymbol{o}$, let the visible keypoints' index set be $\mathcal{M}_{\boldsymbol{o}}$. In other words, $\boldsymbol{o}$ is able to observe $\boldsymbol{p}_{2\mathrm{D},i}$, $i\in\mathcal{M}_{\boldsymbol{o}}$, and therefore $\boldsymbol{o}$ lies within the non-empty covisibility region of $\sum_{k=1}^K c_k\boldsymbol{b}_i^k$, $i\in\mathcal{M}_{\boldsymbol{o}}$. In the following, we assume that whenever a camera center $\boldsymbol{o}$ observes a 3D point $\sum_{k=1}^K c_k\boldsymbol{b}_i^k$, it is also in the visibility region of $\boldsymbol{b}_i^k$ for $i\in\mathcal{M}_{\boldsymbol{o}}$ and for $k=1,\dots K$.

Given an arbitrary triplet of keypoints $\boldsymbol{p}_{2\mathrm{D},i}$, $\boldsymbol{p}_{2\mathrm{D},j}$, and $\boldsymbol{p}_{2\mathrm{D},m}$ (corresponding to $\sum_{k=1}^K c_k\boldsymbol{b}_i^k$, $\sum_{k=1}^K c_k\boldsymbol{b}_j^k$, and $\sum_{k=1}^K c_k\boldsymbol{b}_m^k$), we want to prove (*cf.* Proposition 16)

$$
\boldsymbol{f}_{2D}(\boldsymbol{p}_{2\mathrm{D},i},\boldsymbol{p}_{2\mathrm{D},j},\boldsymbol{p}_{2\mathrm{D},m})\in\boldsymbol{F}_{2D}(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j,\boldsymbol{\theta}_m) \qquad \text{(A24)}
$$

with

$$
\begin{aligned}
&\boldsymbol{f}_{2D}(\boldsymbol{p}_{2\mathrm{D},i},\boldsymbol{p}_{2\mathrm{D},j},\boldsymbol{p}_{2\mathrm{D},m}) \\
&\qquad\doteq\det\left(\begin{bmatrix}\boldsymbol{p}_{2\mathrm{D},j}-\boldsymbol{p}_{2\mathrm{D},i} & \boldsymbol{p}_{2\mathrm{D},m}-\boldsymbol{p}_{2\mathrm{D},i}\end{bmatrix}\right) && \text{(A25)} \\
&\boldsymbol{F}_{2D}(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j,\boldsymbol{\theta}_m)=\bigcup_{k=1}^K\mathcal{W}_k(i,j,m) && \text{(A26)}
\end{aligned}
$$

We have the following cases:

Case 1. $\bigcup_{k=1}^K\mathcal{W}_k(i,j,m)=\{+1,-1\}$. In this case, $\boldsymbol{f}_{2D}(\boldsymbol{p}_{2\mathrm{D},i},\boldsymbol{p}_{2\mathrm{D},j},\boldsymbol{p}_{2\mathrm{D},m})\in\boldsymbol{F}_{2D}(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j,\boldsymbol{\theta}_m)$ trivially since the points are assumed in generic position and hence the determinant is non-zero (*i.e.,* either positive or negative).

Case 2. $\bigcup_{k=1}^K\mathcal{W}_k(i,j,m)=\{+1\}$ or $\bigcup_{k=1}^K\mathcal{W}_k(i,j,m)=\{-1\}$. In other words, the winding order dictionaries $\mathcal{W}_k(i,j,m)$ for all k have the same winding order. To have $\boldsymbol{f}_{2D}(\boldsymbol{p}_{2\mathrm{D},i},\boldsymbol{p}_{2\mathrm{D},j},\boldsymbol{p}_{2\mathrm{D},m})\in\boldsymbol{F}_{2D}(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j,\boldsymbol{\theta}_m)$, we need to find a condition under which the convex combination of keypoints has the same winding order as any of the corresponding keypoints in each individual shape. In other words, we need:

$$
\operatorname{sgn}\left((\boldsymbol{o}-\textstyle\sum_{k=1}^K c_k\boldsymbol{b}_i^k)\cdot\boldsymbol{n}\right)=\mathcal{W}_k(i,j,m) \qquad \text{(A27)}
$$

where $\boldsymbol{n}=(\sum_{k=1}^K c_k(\boldsymbol{b}_j^k-\boldsymbol{b}_i^k))\times(\sum_{k=1}^K c_k(\boldsymbol{b}_m^k-\boldsymbol{b}_i^k))$.

We can rewrite eq. (A27) in a way that eliminates $\boldsymbol{n}$. Towards this goal, we start by using Lemma A2 to establish

the following equality for a triplet of points $\boldsymbol{b}_i^k, \boldsymbol{b}_j^k, \boldsymbol{b}_m^k$ and for each shape:

$$\operatorname{sgn}\left((\boldsymbol{o}-\boldsymbol{b}_i^k)\cdot\boldsymbol{n}_{i,j,m}^k\right)=\operatorname{sgn}\left(\det\begin{bmatrix}\boldsymbol{v}_i^k & \boldsymbol{b}_{ji}^k & \boldsymbol{b}_{mi}^k\end{bmatrix}\right) \quad \forall k=1,\dots,K \tag{A28}$$

where $\boldsymbol{v}_i^k=\boldsymbol{o}-\boldsymbol{b}_i^k$, $\boldsymbol{b}_{ji}^k=\boldsymbol{b}_j^k-\boldsymbol{b}_i^k$, and $\boldsymbol{b}_{mi}^k=\boldsymbol{b}_m^k-\boldsymbol{b}_i^k$ (recall that $\boldsymbol{n}_{i,j,m}^k=(\boldsymbol{b}_j^k-\boldsymbol{b}_i^k)\times(\boldsymbol{b}_m^k-\boldsymbol{b}_i^k)$ from eq. (20)).

Let $\boldsymbol{A}^k=\begin{bmatrix}\boldsymbol{v}_i^k & \boldsymbol{b}_{ji}^k & \boldsymbol{b}_{mi}^k\end{bmatrix}$ for $k=1,\dots,K$. It follows

$$\begin{aligned}&\det\left(\sum_{k=1}^K c_k\boldsymbol{A}^k\right)\\ &=\det\begin{bmatrix}\sum_{k=1}^K c_k\boldsymbol{o}-\sum_{k=1}^K c_k\boldsymbol{b}_i^k & \sum_{k=1}^K c_k\boldsymbol{b}_{ji}^k & \sum_{k=1}^K c_k\boldsymbol{b}_{mi}^k\end{bmatrix}\\ &=\det\begin{bmatrix}\boldsymbol{o}-\sum_{k=1}^K c_k\boldsymbol{b}_i^k & \sum_{k=1}^K c_k\boldsymbol{b}_{ji}^k & \sum_{k=1}^K c_k\boldsymbol{b}_{mi}^k\end{bmatrix}\\ &=(\boldsymbol{o}-\sum_{k=1}^K \boldsymbol{c}_k\boldsymbol{b}_i^k)\cdot\boldsymbol{n}\end{aligned} \tag{A29}$$

Therefore, as long as:

$$\operatorname{sgn}\left(\det\left(\sum_{k=1}^K c_k\boldsymbol{A}^k\right)\right)=\operatorname{sgn}\left(\det\left(\boldsymbol{A}^k\right)\right),\quad \forall\boldsymbol{c}\in\Delta_K \tag{A30}$$

then, we have

$$\begin{aligned}\operatorname{sgn}\left(\det\left(\sum_{k=1}^K c_k\boldsymbol{A}^k\right)\right)&=\operatorname{sgn}\left((\boldsymbol{o}-\sum_{k=1}^K \boldsymbol{c}_k\boldsymbol{b}_i^k)\cdot\boldsymbol{n}\right)\\ &=\mathcal{W}_k(i,j,m)\end{aligned} \tag{A31}$$

which matches the claim in Proposition 16 since it implies $\boldsymbol{f}_{2D}(\boldsymbol{p}_{2D,i},\boldsymbol{p}_{2D,j},\boldsymbol{p}_{2D,m})\in\boldsymbol{F}_{2D}(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j,\boldsymbol{\theta}_m)$.

While condition (A30) might not be satisfied in general, with simulated data, we check empirically whether Proposition 16 and eq. (A31) hold under scenarios where the ground-truth shape coefficients are randomly sampled in the probability simplex Δ_K. Fig. A3 shows the percentage of test instances where eq. (A31) holds true under the same synthetic data generation procedure for the robustness experiments described in Section VIII-B, with octahedra as shapes. We set the outlier rate to 0% (the invariant is only expected to hold for inliers), while changing the number of shapes from 2 to 5. At all shape counts, eq. (A31) holds true for 100% of the generated test instances.

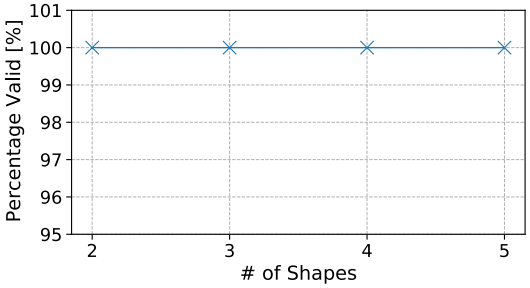


Fig. A3. Percentage of simulated test instances where eq. (A31) and Proposition 16 hold even when the keypoints are generated by linear combinations of shapes (rather than by one of the shapes in the CAD library).

APPENDIX H
GENERATING WINDING ORDER DICTIONARIES

In this section, we discuss three different methods for constructing winding order dictionaries. The first method uses linear programming to solve eq. (22) and eq. (23) directly. It is suitable for cases where analytical equations of the faces of the CAD models are known, and the shapes are convex. The second method, which uses ray tracing, is applicable to complex non-convex shapes. The last method constructs the winding order dictionaries by learning from ground-truth 2D annotations. Our simulated experiments for PACE2D# uses the first method. Our experiments on ApolloScape uses a combination of the last two methods.

A. Using Linear Programs

Eq. (22) and eq. (23) are two feasibility problems that can be solved using linear program solvers, if $\boldsymbol{o}\in\mathcal{C}$ can be expressed as linear constraints. Luckily, this can be achieved if the underlying shape is closed and convex, with keypoints lying strictly in the interior of the faces (excluding the vertices and edges). In essence, for keypoints on an arbitrary face, their visibility regions are equivalent to the half-space outside the shape. Assume we have L faces with known plane equations

$$\boldsymbol{n}_l\cdot\boldsymbol{o}+b_l=0,\quad \forall l=1,\dots,L, \tag{A32}$$

where $\boldsymbol{n}_l$ is the normal vector of face l pointing *away* from the model, and b_l is a constant. Then, the visibility region of all keypoints on face l is equivalent to checking whether there exists a $\boldsymbol{o}$ such that

$$\boldsymbol{n}_l\cdot\boldsymbol{o}+b_l>0 \tag{A33}$$

The covisibility region of a keypoint triplet i,j,m can therefore be found by combining the above constraints for all three keypoints

$$\boldsymbol{n}_{l_1}\cdot\boldsymbol{o}+b_{l_1}>0 \tag{A34}$$
$$\boldsymbol{n}_{l_2}\cdot\boldsymbol{o}+b_{l_2}>0 \tag{A35}$$
$$\boldsymbol{n}_{l_3}\cdot\boldsymbol{o}+b_{l_3}>0 \tag{A36}$$

where l_1, l_2, and l_3 are the indices of the faces keypoints i, j, and m belong to. Fig. A4 shows an example of a covisibility region of three keypoints on a cube, which is the intersection of two half-spaces.

The feasibility linear program for solving eq. (22) is then

$$\begin{aligned}\text{find}\quad & \boldsymbol{o}\\ \text{subject to}\quad & \boldsymbol{n}_{l_1}\cdot\boldsymbol{o}+b_{l_1}\geq C\\ & \boldsymbol{n}_{l_2}\cdot\boldsymbol{o}+b_{l_2}\geq C\\ & \boldsymbol{n}_{l_3}\cdot\boldsymbol{o}+b_{l_3}\geq C\\ & (\boldsymbol{o}-\boldsymbol{b}_i^k)\cdot\boldsymbol{n}_{i,j,m}^k\geq C\end{aligned} \tag{A37}$$

where C is a small positive constant. If (A37) has a solution, that means keypoints i,j,m can be viewed in counterclockwise winding order. And the feasibility linear program for solving eq. (22) is

$$\begin{aligned}\text{find}\quad & \boldsymbol{o}\\ \text{subject to}\quad & \boldsymbol{n}_{l_1}\cdot\boldsymbol{o}+b_{l_1}\geq C\\ & \boldsymbol{n}_{l_2}\cdot\boldsymbol{o}+b_{l_2}\geq C\\ & \boldsymbol{n}_{l_3}\cdot\boldsymbol{o}+b_{l_3}\geq C\\ & (\boldsymbol{o}-\boldsymbol{b}_i^k)\cdot\boldsymbol{n}_{i,j,m}^k\leq -C\end{aligned} \tag{A38}$$

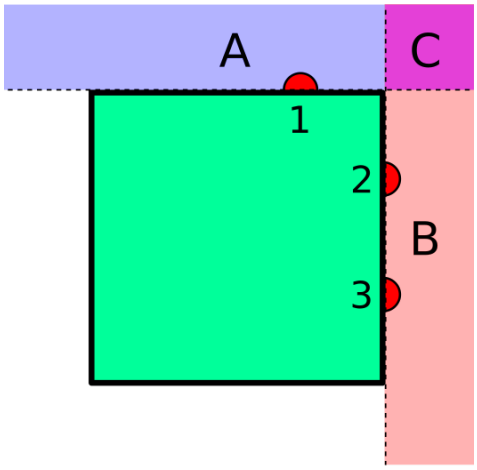


Fig. A4. A top down view of a cube, with 3 keypoints (1, 2, and 3) on two different faces. Shaded region A (*blue*) is the visibility region of keypoint 1. Shaded region B (*red*) is the visibility region of 2 and 3. Shaded region C (*purple*) is the covisibility region of 1, 2, and 3.

If (A38) has a solution, that means keypoints i, j, m can be viewed in clockwise winding order. To construct the winding order dictionary, we simply solve (A37) and (A37) for all triplets, and record the results.

B. Using Ray Tracing

For shapes where we do not have access to analytical face equations, or non-convex shapes, ray tracing can be used instead. The main insight is to use ray tracing to check visibilities of keypoints from a set of sampled camera locations around the model, instead of using linear programs.

Given a shape model, we first discretize the volume surrounding it into voxels, with each voxel center outside the model representing a potential optical center for ray tracing. For each optical center, we perform ray tracing with all keypoints to determine the set of visible keypoints. We store the results of ray tracing into a dictionary of 3D boolean arrays, with keypoints as keys. For any keypoint, its boolean array represents the visibility of it from each optical center.

After we have the dictionary, for an arbitrary triplet of keypoints i, j and m, we can check whether the constraint $\boldsymbol{o} \in \mathcal{C}$ can be met by performing an `AND` operation on the boolean visibility arrays of the three keypoints. If the resulting array after `AND` has at least one `True` value, that means there exists one optical center such that all three keypoints are visible. Hence $\boldsymbol{o} \in \mathcal{C}$ can be satisfied. The feasibility problems of eq. (22) and eq. (23) can then be checked by calculating $(\boldsymbol{o} - \boldsymbol{b}_i^k) \cdot \boldsymbol{n}_{i,j,m}^k$ for all optical centers in $\mathcal{C}$.

C. Learning From 2D Annotations

In cases where ground-truth 2D keypoints annotations are available, we may also construct winding order dictionaries by learning. We first construct a dictionary D with keypoint triplets as keys and empty sets as values. For each ground-truth annotated training image, we calculate the winding order following Proposition 10 for each triplet (i, j, m) in the image, and push the result into the (i, j, m) entry of D. Since the values are sets, only unique winding orders will be preserved. After enumerating through all the training images, if there are triplets that were not found during training, we use ray tracing to estimate the feasible winding orders. We use this approach to construct the winding order dictionary for our experiments on the ApolloScape dataset.

Appendix I
Problem (3D-3D) is a MAP Estimator when the Measurement Noise is Gaussian

Here we prove that the optimization in eq. (3D-3D) is a *maximum a posteriori* (MAP) estimator when the measurement noise $\boldsymbol{\epsilon}_{3\mathrm{D},i}$ in (2) follows a zero-mean Gaussian with covariance $\frac{1}{w_i}\mathbf{I}_3$ (where $\mathbf{I}_3$ is the 3-by-3 identity matrix) and we have a zero-mean Gaussian prior with covariance $\frac{1}{\lambda}\mathbf{I}_K$ over the shape parameters $\boldsymbol{c}$. Mathematically:

$$\mathbb{P}\left(\boldsymbol{\epsilon}_{3\mathrm{D},i}\right) = \kappa_\epsilon \exp\left(-\frac{w_i}{2}\|\boldsymbol{\epsilon}_{3\mathrm{D},i}\|^2\right), \tag{A39}$$

$$\mathbb{P}\left(\boldsymbol{c}\right) = \kappa_c \exp\left(-\frac{\lambda}{2}\|\boldsymbol{c}\|^2\right), \tag{A40}$$

where κ_ϵ and κ_c are suitable normalization constants that are irrelevant for the following derivation.

A MAP estimator for the unknown parameters $\boldsymbol{x} \triangleq \{\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c}\}$ (belonging to a suitable domain $\mathbb{X}$) given measurements $\boldsymbol{p}_{3\mathrm{D},i}$ $(i = 1, \ldots, N)$ is defined as the maximum of the posterior distribution $\mathbb{P}\left(\boldsymbol{x} | \boldsymbol{p}_{3\mathrm{D},1} \ \ldots \ \boldsymbol{p}_{3\mathrm{D},N}\right)$:

$$\arg\max_{\boldsymbol{x}\in\mathbb{X}} \mathbb{P}\left(\boldsymbol{x} | \boldsymbol{p}_{3\mathrm{D},1} \ \ldots \ \boldsymbol{p}_{3\mathrm{D},N}\right) = \arg\max_{\boldsymbol{x}\in\mathbb{X}} \prod_{i=1}^{N} \mathbb{P}\left(\boldsymbol{p}_{3\mathrm{D},i} | \boldsymbol{x}\right) \mathbb{P}\left(\boldsymbol{x}\right) \tag{A41}$$

where on the right we applied Bayes rule and used the standard assumption of independent measurements. Using (A39) and (2) we obtain:

$$\mathbb{P}\left(\boldsymbol{p}_{3\mathrm{D},i} | \boldsymbol{x}\right) = \kappa_\epsilon \exp\left(-\frac{w_i}{2}\left\|\boldsymbol{p}_{3\mathrm{D},i} - \boldsymbol{R}\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k - \boldsymbol{t}\right\|^2\right) \tag{A42}$$

Moreover, assuming we only have a prior on $\boldsymbol{c}$:

$$\mathbb{P}\left(\boldsymbol{x}\right) = \mathbb{P}\left(\boldsymbol{c}\right) = \kappa_c \exp\left(-\frac{\lambda}{2}\|\boldsymbol{c}\|^2\right). \tag{A43}$$

Substituting (A42) and (A43) back into (A41) and observing that the maximum of the posterior is the same as the minimum of the negative logarithm of the posterior:

$$\arg\max_{\boldsymbol{x}\in\mathbb{X}} \prod_{i=1}^{N} \mathbb{P}\left(\boldsymbol{p}_{3\mathrm{D},i} | \boldsymbol{x}\right) \mathbb{P}\left(\boldsymbol{x}\right) = \tag{A44}$$

$$\arg\min_{\boldsymbol{x}\in\mathbb{X}} \sum_{i=1}^{N} -\log \mathbb{P}\left(\boldsymbol{p}_{3\mathrm{D},i} | \boldsymbol{x}\right) - \log \mathbb{P}\left(\boldsymbol{x}\right) = \tag{A45}$$

$$\arg\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3),\\ \boldsymbol{t}\in\mathbb{R}^3, \boldsymbol{c}\in\mathbb{R}^K,\\ \mathbf{1}^\mathsf{T}\boldsymbol{c}=1}} \sum_{i=1}^{N} \frac{w_i}{2}\left\|\boldsymbol{p}_{3\mathrm{D},i} - \boldsymbol{R}\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k - \boldsymbol{t}\right\|^2 \tag{A46}$$

$$+\frac{\lambda}{2}\|\boldsymbol{c}\|^2 + \text{constants} \tag{A47}$$

which, after dropping constant multiplicative and additive factors, can be seen to match eq. (3D-3D), proving the claim.

APPENDIX J
CERTIFIABLY OPTIMAL ROTATION ESTIMATION: PROOF OF PROPOSITION 18

Let us first develop the cost function of problem (37) as a quadratic function of $\boldsymbol{r} \triangleq \mathrm{vec}\,(\boldsymbol{R})$:

$$\left\|\boldsymbol{M}(\mathbf{I}_N \otimes \boldsymbol{R}^{\mathsf{T}})\bar{\boldsymbol{y}} + \boldsymbol{h}\right\|^2 \tag{A48}$$
$$= \left\|\boldsymbol{M}\mathrm{vec}\left(\boldsymbol{R}^{\mathsf{T}}\boldsymbol{Y}\right) + \boldsymbol{h}\right\|^2 \tag{A49}$$
$$= \left\|\boldsymbol{M}(\boldsymbol{Y}^{\mathsf{T}} \otimes \mathbf{I}_3)\mathrm{vec}\left(\boldsymbol{R}^{\mathsf{T}}\right) + \boldsymbol{h}\right\|^2 \tag{A50}$$
$$= \left\|\boldsymbol{M}(\boldsymbol{Y}^{\mathsf{T}} \otimes \mathbf{I}_3)\boldsymbol{P}\boldsymbol{r} + \boldsymbol{h}\right\|^2 \tag{A51}$$
$$= \tilde{\boldsymbol{r}}^{\mathsf{T}}\boldsymbol{Q}\tilde{\boldsymbol{r}} \tag{A52}$$

where $\boldsymbol{P} \in \mathbb{R}^{9\times 9}$ is the following permutation matrix

$$(1,1,1),(2,4,1),(3,7,1), \tag{A53}$$
$$(4,2,1),(5,5,1),(6,8,1), \tag{A54}$$
$$(7,3,1),(8,6,1),(9,9,1), \tag{A55}$$

with the triplet (i,j,v) defining the nonzero entries of $\boldsymbol{P}$ (*i.e.,* $\boldsymbol{P}_{ij} = v$), such that:

$$\mathrm{vec}\left(\boldsymbol{R}^{\mathsf{T}}\right) \equiv \boldsymbol{P}\mathrm{vec}\,(\boldsymbol{R}) \tag{A56}$$

always holds, $\boldsymbol{Y}$ and $\tilde{\boldsymbol{r}}$ are defined as:

$$\boldsymbol{Y} \triangleq \begin{bmatrix} \bar{\boldsymbol{y}}_1 & \cdots & \bar{\boldsymbol{y}}_N \end{bmatrix} \in \mathbb{R}^{3\times N}, \tag{A57}$$
$$\tilde{\boldsymbol{r}} \triangleq \begin{bmatrix} 1 & \boldsymbol{r}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{10}, \tag{A58}$$

and $\boldsymbol{Q} \in \mathcal{S}^{10}$ can be assembled as follows:

$$\boldsymbol{Q} \triangleq \begin{bmatrix} \boldsymbol{h}^{\mathsf{T}}\boldsymbol{h} & \boldsymbol{h}^{\mathsf{T}}\boldsymbol{M}(\boldsymbol{Y}^{\mathsf{T}} \otimes \mathbf{I}_3)\boldsymbol{P} \\ \star & \boldsymbol{P}^{\mathsf{T}}(\boldsymbol{Y} \otimes \mathbf{I}_3)\boldsymbol{M}^{\mathsf{T}}\boldsymbol{M}(\boldsymbol{Y}^{\mathsf{T}} \otimes \mathbf{I}_3)\boldsymbol{P} \end{bmatrix}. \tag{A59}$$

Now that the objective function of (37) is quadratic in $\boldsymbol{r}$ ($\boldsymbol{R}$), we can write problem (37) equivalently as the *quadratically constrained quadratic program* (QCQP) in (39), where $\boldsymbol{A}_i \in \mathcal{S}^{10}, i = 1,\ldots,15$, are the constant matrices that define the quadratic constraints associated with $\boldsymbol{R} \in \mathrm{SO}(3)$ [117, Lemma 5]. For completeness, we give the expressions for $\boldsymbol{A}_i$'s:

$$\boldsymbol{A}_0 : (1,1,1)$$
$$\boldsymbol{A}_1 - \boldsymbol{A}_3 : \text{ columns have unit norm}$$
$$\boldsymbol{A}_1 : (1,1,1),(2,2,-1),(3,3,-1),(4,4,-1)$$
$$\boldsymbol{A}_2 : (1,1,1),(5,5,-1),(6,6,-1),(7,7,-1)$$
$$\boldsymbol{A}_3 : (1,1,1),(8,8,-1),(9,9,-1),(10,10,-1)$$

$$\boldsymbol{A}_4 - \boldsymbol{A}_6 : \text{ columns are mutually orthogonal}$$
$$\boldsymbol{A}_4 : (2,5,1),(3,6,1),(4,7,1)$$
$$\boldsymbol{A}_5 : (2,8,1),(3,9,1),(4,10,1)$$
$$\boldsymbol{A}_6 : (5,8,1),(6,9,1),(7,10,1)$$

$$\boldsymbol{A}_7 - \boldsymbol{A}_{15} : \text{ columns form right-handed frame}$$
$$\boldsymbol{A}_7 : (3,7,1),(4,6,-1),(1,8,-1)$$
$$\boldsymbol{A}_8 : (4,5,1),(2,7,-1),(1,9,-1)$$
$$\boldsymbol{A}_9 : (2,6,1),(1,10,-1),(3,5,-1)$$
$$\boldsymbol{A}_{10} : (6,10,1),(1,2,-1),(7,9,-1)$$
$$\boldsymbol{A}_{11} : (7,8,1),(5,10,-1),(1,3,-1)$$
$$\boldsymbol{A}_{12} : (5,9,1),(1,4,-1),(6,8,-1)$$
$$\boldsymbol{A}_{13} : (4,9,1),(3,10,-1),(1,5,-1)$$
$$\boldsymbol{A}_{14} : (2,10,1),(1,6,-1),(4,8,-1)$$
$$\boldsymbol{A}_{15} : (3,8,1),(2,9,-1),(1,7,-1)$$

where the triplets (i,j,v) define the *diagonal and upper triangular* nonzero entries of a symmetric matrix (*i.e.,* $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji} = v$ with $i \le j$).

APPENDIX K
SHOR'S SEMIDEFINITE RELAXATION: PROOF OF PROPOSITION 18

Proof: To see why problem (40) is a convex relaxation for problem (39), let us first create a matrix variable

$$\boldsymbol{X} = \tilde{\boldsymbol{r}}\tilde{\boldsymbol{r}}^{\mathsf{T}} \in \mathcal{S}^{10}, \tag{A60}$$

and notice that $\boldsymbol{X}$ satisfies

$$\boldsymbol{X} \succeq 0, \quad \mathrm{rank}\,(\boldsymbol{X}) = 1. \tag{A61}$$

Moreover, if $\boldsymbol{X} \succeq 0, \mathrm{rank}\,(\boldsymbol{X}) = 1$ then $\boldsymbol{X}$ must have a factorization of the form (A60). Therefore, the non-convex QCQP (39) is equivalent to the following rank-constrained matrix optimization problem:

$$\min_{\boldsymbol{X}\in\mathcal{S}^{10}} \quad \mathrm{tr}\,(\boldsymbol{Q}\boldsymbol{X}) \tag{A62}$$
$$\text{s.t.} \quad \mathrm{tr}\,(\boldsymbol{A}_0\boldsymbol{X}) = 1, \tag{A63}$$
$$\mathrm{tr}\,(\boldsymbol{A}_i\boldsymbol{X}) = 0, \forall i = 1,\ldots,15, \tag{A64}$$
$$\boldsymbol{X} \succeq 0, \tag{A65}$$
$$\mathrm{rank}\,(\boldsymbol{X}) = 1, \tag{A66}$$

where $\boldsymbol{A}_0 \in \mathcal{S}^{10}$ is an all-zero matrix except the top-left entry being 1 (to enforce that the first entry of $\tilde{\boldsymbol{r}}$ is 1), and we have used the fact that

$$\tilde{\boldsymbol{r}}^{\mathsf{T}}\boldsymbol{A}\tilde{\boldsymbol{r}} = \mathrm{tr}\left(\tilde{\boldsymbol{r}}^{\mathsf{T}}\boldsymbol{A}\tilde{\boldsymbol{r}}\right) = \mathrm{tr}\left(\boldsymbol{A}\tilde{\boldsymbol{r}}\tilde{\boldsymbol{r}}^{\mathsf{T}}\right) = \mathrm{tr}\,(\boldsymbol{A}\boldsymbol{X}). \tag{A67}$$

Now observe that the only nonconvex constraint in problem (A62) is the rank constraint (A66), and the SDP relaxation (40) is obtained by simply removing the rank constraint. ■

In practice, we solve the convex problem (40) and obtain an optimal solution $\boldsymbol{X}^\star$; if $\mathrm{rank}\,(\boldsymbol{X}^\star) = 1$, then the optimal solution of problem (40) is unique (the rationale behind this is that interior-point methods converge to a maximum rank solution [20]) and it actually satisfies the rank constraint that has been dropped. Therefore, in this situation, we say the convex relaxation is tight and the global optimal solution to the nonconvex problem (39) can be obtained from the rank-one factorization of $\boldsymbol{X}^\star$.

APPENDIX L
ROUNDING AND SUBOPTIMALITY GAP FOR (40)

This appendix discusses how to compute a feasible rotation estimate and the corresponding suboptimality gap from the solution $\boldsymbol{X}^\star$ of the SDP relaxation (40). Let $f^\star$ be the optimal objective value of the SDP (40), and $\boldsymbol{X}^\star = \sum_{i=1}^{10} \gamma_i \boldsymbol{u}_i \boldsymbol{u}_i^\mathsf{T}$ be the spectral decomposition of $\boldsymbol{X}^\star$ with $\gamma_1 \geq \ldots \geq \gamma_{10}$. We define a rounding procedure

$$\boldsymbol{u}_i \leftarrow \frac{\boldsymbol{u}_1}{\boldsymbol{u}_1(1)}, \quad \widehat{\boldsymbol{R}} = \text{proj}_{\text{SO}(3)}(\boldsymbol{u}_1(\boldsymbol{r})) \tag{A68}$$

that extracts a feasible point $\widehat{\boldsymbol{R}}$ to (39) from the leading eigenvector $\boldsymbol{u}_1$ of $\boldsymbol{X}^\star$. In (A68), $\boldsymbol{u}_1(\cdot)$ extracts the entries of $\boldsymbol{u}_1$ using the indices of "$\cdot$" in $\tilde{\boldsymbol{r}}$, and $\text{proj}_{\text{SO}(3)}$ represents the projection onto $\text{SO}(3)$. Given $\widehat{\boldsymbol{R}}$, denote the objective value of (39) at $\widehat{\boldsymbol{R}}$ as $\widehat{p}$; we compute a *relative suboptimality* as

$$\eta = |\widehat{p} - f^\star| \, / (1 + |\widehat{p}| + |f^\star|) \tag{A69}$$

to evaluate the quality of the feasible solution. Apparently, $\eta = 0$ certifies the global optimality of $\widehat{\boldsymbol{R}}$.

APPENDIX M
ALTERNATION APPROACH

In Sections VI-A2-VI-A3 of the main paper, we presented a certifiably optimal solver to solve the shape and rotation $(\boldsymbol{c}, \boldsymbol{R})$ problem (30) (after eliminating the translation $\boldsymbol{t}$). Here we describe a baseline method that solves problem (30) using *alternating minimization* (Altern), a heuristic that is popular in related works on 3D shape reconstruction from 2D landmarks [56], [34], [77], but offers no optimality guarantees. Towards this goal, let us denote the cost function of (30) as $f(\boldsymbol{R}, \boldsymbol{c})$; the Altern method starts with an initial guess $(\boldsymbol{R}^{(0)}, \boldsymbol{c}^{(0)})$ (default $\boldsymbol{R}^{(0)} = \mathbf{I}_3, \boldsymbol{c}^{(0)} = \mathbf{0}$), and performs the following two steps at each iteration τ:

1) Optimize $\boldsymbol{c}$:

$$\boldsymbol{c}^{(\tau)} = \underset{\boldsymbol{c} \in \mathbb{R}^K, \mathbf{1}^\mathsf{T}\boldsymbol{c}=1}{\arg\min} \; f(\boldsymbol{R}^{(\tau-1)}, \boldsymbol{c}), \tag{A70}$$

which is a linearly constrained linear least squares problem and can be solved by the closed-form solution (34).

2) Optimize $\boldsymbol{R}$:

$$\boldsymbol{R}^{(\tau)} = \underset{\boldsymbol{R} \in \text{SO}(3)}{\arg\min} \, f(\boldsymbol{R}, \boldsymbol{c}^{(\tau)}), \tag{A71}$$

which can be cast as an instance of Wahba's problem [116] and can be solved in closed form using singular value decomposition [61].

The Altern method stops when the cost function converges, *i.e.,* $|f(\boldsymbol{R}^{(\tau)}, \boldsymbol{c}^{(\tau)}) - f(\boldsymbol{R}^{(\tau-1)}, \boldsymbol{c}^{(\tau-1)})| < \epsilon$ for some small threshold $\epsilon > 0$, or when τ exceeds the maximum number of iterations (*e.g.,* 1000).

APPENDIX N
ROUNDING AND SUBOPTIMALITY GAP FOR (45)

This section provides extra results related to Lasserre's Hierarchy of semidefinite relaxations and provides a rounding procedure to obtain a rotation and shape parameters estimate from the solution of the SDP (45).

Corollary A3 (Optimality Certification from Lasserre's Hierarchy [49]). *Let $p^\star$ and $f^\star$ be the optimal objectives of (43) and (45), respectively, and let $\boldsymbol{X}^\star = (\boldsymbol{X}_0^\star, \ldots, \boldsymbol{X}_K^\star)$ be an optimal solution of (45), we have*

(i) $f^\star \leq p^\star$,

(ii) if $\text{rank}\,(\boldsymbol{X}_0^\star) = 1$, then $f^\star = p^\star$, and $\boldsymbol{X}_0^\star$ can be factorized as $\boldsymbol{X}_0^\star = [\boldsymbol{x}^\star][\boldsymbol{x}^\star]_2^\mathsf{T}$, where $\boldsymbol{x}^\star = [\text{vec}\,(\boldsymbol{R}^\star)^\mathsf{T}, (\boldsymbol{c}^\star)^\mathsf{T}]^\mathsf{T}$ is a globally optimal for (43).

Rounding. Empirically, we observe that solving the SDP (45) empirically yields a rank-one optimal solution, and hence it typically allows retrieving the global solution of the non-convex problem (43) per Corollary A3. Even when the SDP solution is not rank-one, we can "round" a feasible solution to (43) from $\boldsymbol{X}_0^\star$. To do so, let $\boldsymbol{X}_0^\star = \sum_{i=1}^{n_0} \gamma_i \boldsymbol{u}_i \boldsymbol{u}_i^\mathsf{T}$ be the spectral decomposition of $\boldsymbol{X}_0^\star$ with $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_{n_0}$. To extract a feasible point $(\widehat{\boldsymbol{R}}, \widehat{\boldsymbol{c}})$ for (43) from the leading eigenvector $\boldsymbol{u}_1$, we follow

$$\boldsymbol{u}_1 \leftarrow \frac{\boldsymbol{u}_1}{\boldsymbol{u}_1(1)}, \widehat{\boldsymbol{R}} = \text{proj}_{\text{SO}(3)}\left(\boldsymbol{u}_1(\boldsymbol{r})\right), \widehat{\boldsymbol{c}} = \text{proj}_{\Delta_K}\left(\boldsymbol{u}_1(\boldsymbol{c})\right)$$

where proj_{Δ_K} represents the projection onto Δ_K [111]. We can then evaluate the relative suboptimality of the rotation and shape estimate $(\widehat{\boldsymbol{R}}, \widehat{\boldsymbol{c}})$ using (A69).

APPENDIX O
2D-3D CATEGORY-LEVEL PERCEPTION WITH ONE-HOT SHAPE VECTOR

Lasserre's relaxation allows tackling even harder problem instances and constraints compared to (43). For instance, we can easily enforce the optimization to select a single shape (*i.e.,* finding a one-hot shape vector $\boldsymbol{c}$). This is equivalent to solving the following variant of (43):

$$\min_{\substack{\boldsymbol{R} \in \text{SO}(3) \\ \boldsymbol{c} \in \{0,1\}^K, \mathbf{1}^\mathsf{T}\boldsymbol{c}=1}} \sum_{i=1}^{N} \left\| \boldsymbol{R}\boldsymbol{s}_i(\boldsymbol{c}) - \sum_{j=1}^{N} \widetilde{\boldsymbol{W}}_j \boldsymbol{R}\boldsymbol{s}_j(\boldsymbol{c}) \right\|_{\boldsymbol{W}_i}^2 \tag{A72}$$

where the constraints $\boldsymbol{c} \in \{0,1\}^K, \mathbf{1}^\mathsf{T}\boldsymbol{c} = 1$ imply $\boldsymbol{c}$ is a one-hot vector. One can observe that problem (A72) is also a POP because the constraint $c \in \{0,1\}^K$ can be written as polynomial equality constraints $c_k(c_k - 1) = 0, k = 1, \ldots, K$. Therefore, we can relax (A72) to an SDP (45), while retaining the same optimality guarantees of Corollary A3.[6]

APPENDIX P
GRADUATED NON-CONVEXITY FOR CATEGORY-LEVEL PERCEPTION

Robust 3D-3D Category-level Perception. As prescribed by standard robust estimation, we can re-gain robustness to outliers by replacing the squared ℓ_2 norm in (3D-3D) with a robust loss function ρ, leading to

$$\min_{\substack{\boldsymbol{R} \in \text{SO}(3), \\ \boldsymbol{t} \in \mathbb{R}^3, \boldsymbol{c} \in \Delta_K}} \sum_{i=1}^{N} \rho\left(\left\| \boldsymbol{p}_{3\text{D},i} - \boldsymbol{R} \sum_{k=1}^{K} c_k \boldsymbol{b}_i^k - \boldsymbol{t} \right\| \right) + \lambda \left\| \boldsymbol{c} \right\|^2 \tag{A73}$$

[6]The rounding procedure (A72) needs to be modified accordingly: $\widehat{\boldsymbol{c}}$ is obtained by setting the largest entry of $\boldsymbol{u}_1(\boldsymbol{c})$ to 1 and the remaining entries to 0.

While GNC can be applied to a broad class of loss functions [115], here we consider a truncated least square loss $\rho(r) = \min(r^2, \beta_{3D}^2)$ which minimizes the squared residuals whenever they are below β_{3D}^2 or becomes constant otherwise (note: the constant β_{3D} is the same inlier threshold of Section V-A). Such cost function can be written by using auxiliary slack variables $\rho(r) = \min(r^2, \beta_{3D}^2) = \min_{\omega\in\{0,1\}} \omega r^2 + (1-\omega)\beta_{3D}^2$ [115], hence allowing to rewrite (A73) as

$$\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3),\\ \boldsymbol{t}\in\mathbb{R}^3,\boldsymbol{c}\in\Delta_K\\ \omega_i\in\{0,1\}\forall i}} \sum_{i=1}^{N} \omega_i \left\| \boldsymbol{p}_{3\mathrm{D},i} - \boldsymbol{R}\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k - \boldsymbol{t} \right\|^2 + (1-\omega_i)\beta_{3D}^2 + \lambda \left\|\boldsymbol{c}\right\|^2 \quad \text{(A74)}$$

In (A74), when $\omega_i = 1$, the i-th measurement is considered an inlier and the cost minimizes the corresponding squared residual; when $\omega_i = 0$, the cost becomes independent of $\boldsymbol{p}_{3\mathrm{D},i}$ hence the corresponding measurement is rejected as an outlier. Therefore, equation (A74) simultaneously estimates pose and shape variables $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$ while classifying inliers/outliers via the binary weights ω_i $(i = 1, \ldots, N)$. Now the advantage is that we can minimize (A74) with an alternation scheme where we iteratively optimize (i) over $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$ with fixed weights ω_i and (ii) over the weights ω_i with fixed $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$. This approach is convenient since the optimization over $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$ can be solved to optimality with PACE3D* (see Section VI-A), while the optimization of the weights can be solved in closed form [115]. To improve convergence of this alternation scheme, we adopt graduated non-convexity [9], [115], which starts with a convex approximation of the loss function in (A74) and then gradually increases the non-convexity until the loss ρ in (A74) is recovered.

Robust 2D-3D Category-level Perception. Similarly, we adopt the graduate non-convexity scheme to robustify our solver for Problem 2. We reformulate (41) with a truncated least square loss cost function $\rho(r) = \min(r^2, \beta_{2D}^2)$ into

$$\min_{\substack{\boldsymbol{R}\in\mathrm{SO}(3),\\ \boldsymbol{t}\in\mathbb{R}^3,\boldsymbol{c}\in\Delta_K\\ \omega_i\in\{0,1\}\forall i}} \sum_{i=1}^{N} \omega_i \left\| \boldsymbol{p}_{2\mathrm{D},i} - \pi\left(\boldsymbol{R}\textstyle\sum_{k=1}^{K} c_k \boldsymbol{b}_i^k + \boldsymbol{t}\right) \right\|^2 + (1-\omega_i)\beta_{2D}^2 + \lambda \left\|\boldsymbol{c}\right\|^2 \quad \text{(A75)}$$

To minimize (A75), we adopt the same alternation scheme as for (A74), where we iteratively optimize (i) over $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$ with fixed weights ω_i and (ii) over the weights ω_i with fixed $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$. To tackle the optimization over $(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{c})$, we generate an initial solution by solving (2D-3D) using PACE2D*, and then locally refine the solution using the geometric reprojection error in (41). In Appendix Q-B we show the local refinement improves the quality of the resulting estimates.

APPENDIX Q
EXTRA EXPERIMENTAL RESULTS

A. Results on PACE3D* and PACE3D#

In Section VIII-A, we demonstrate the robustness of PACE3D# to 92% outlier rates when $N = 100$, $K = 10$, and $r = 0.1$. Here we show extra results when K and r are increased. Fig. A5(a) shows the results for $N = 100$, $K = 10$, and $r = 0.2$. One can see that as the intra-class variation radius r is increased, the compatibility checks become less effective, leading to a slight decrease in the robustness of PACE3D# against outliers; PACE3D# is still robust to up to 90% outliers while has two failures at 91% outliers. However, PACE3D# still outperforms IRLS-TLS and IRLS-GM by a large margin. Fig. A5(b) shows the results for $N = 100$, $K = 50$, and $r = 0.1$. We see that PACE3D# is robust to 91% outliers while encounters two failures at 92% outlier rate. Finally, when $K = 50$ and $r = 0.2$ (Fig. A5(c)), PACE3D# is robust to 80% outlier rate.

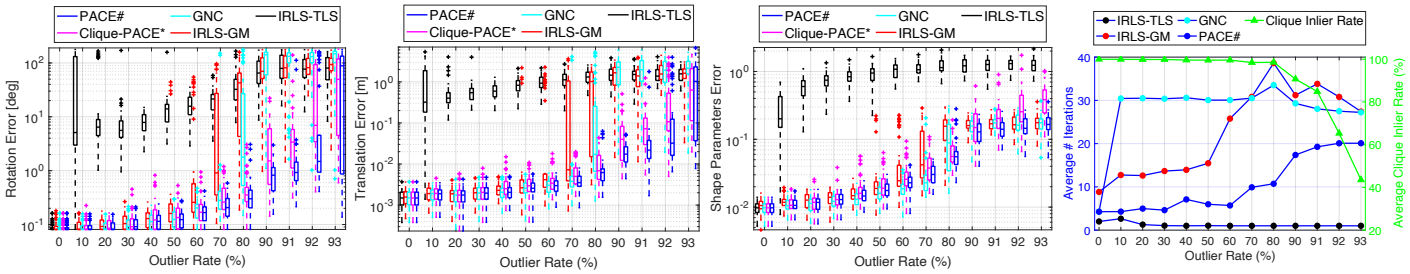
B. Results on PACE2D* and PACE2D#

In Section VIII-B, we demonstrate the robustness of PACE2D# (OH) to 20% outlier rates when $N = 10$ and $K = 3$, with shape coefficients $\boldsymbol{c}$ sampled as random one-hot vectors. Fig. A6 (a) shows extra results of PACE2D# when $\boldsymbol{c}$ are sampled from Δ_K uniformly at random. We see PACE2D# remains robust at 20%, showing failures at 30%. Notably, PACE2D# achieves zero shape errors from $0 - 20\%$ outlier rates.

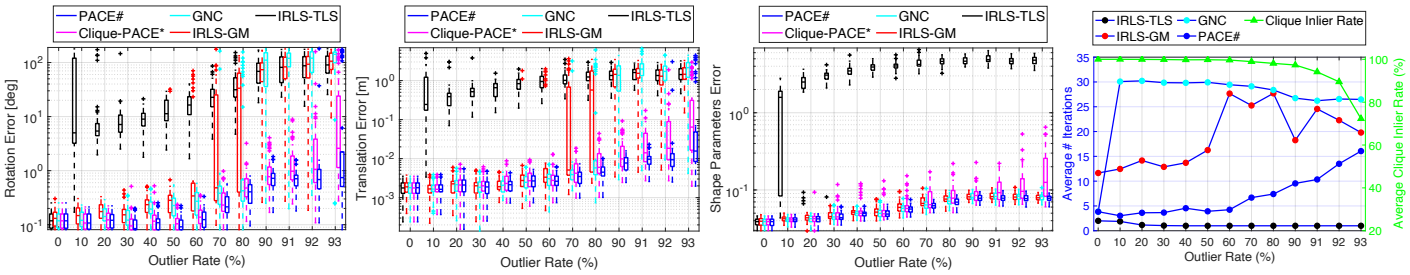
Fig. A6 (b) shows GNC-PACE2D* running with and without local refinement minimizing the reprojection cost. Evidently, GNC-PACE2D* without refinement has consistently more failure cases, especially in terms of shape errors. Note that using refinement incurs a higher runtime cost, as in each GNC iteration we need to run additional iterations of the local solver minimizing the reprojection error.

C. Visualization of Results on ApolloScape

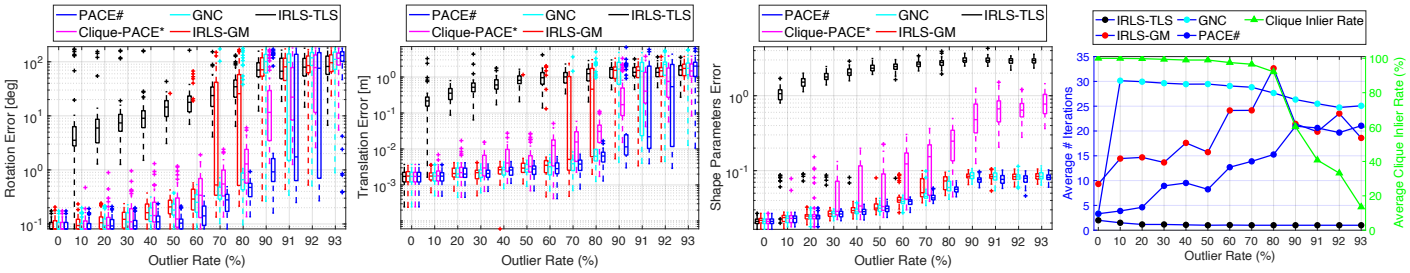
Fig. A6 shows results obtained with PACE3D# on ApolloScape. Fig. A6 shows results obtained with PACE2D# on ApolloScape. In both figures, first four rows show successful results, while the last row demonstrates failure cases.



(a) Robustness of PACE3D# against increasing outliers on synthetic data: $N = 100$, $K = 10$, $r = 0.2$.

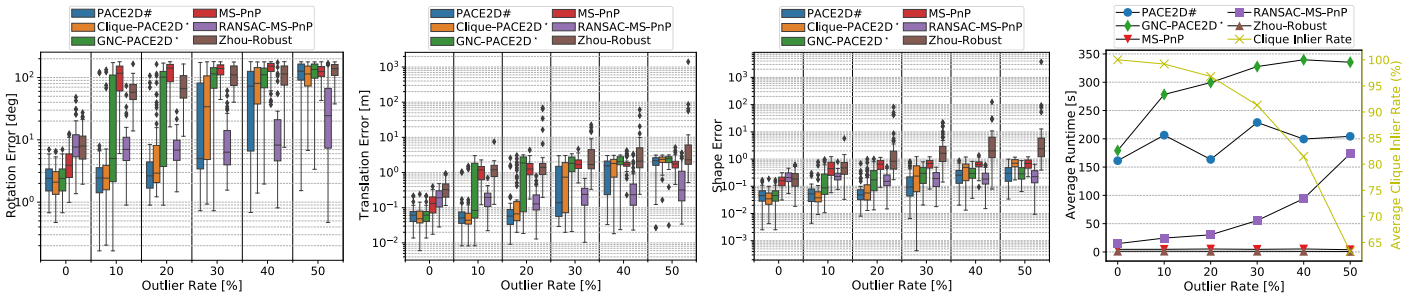


(b) Robustness of PACE3D# against increasing outliers on synthetic data: $N = 100$, $K = 50$, $r = 0.1$.

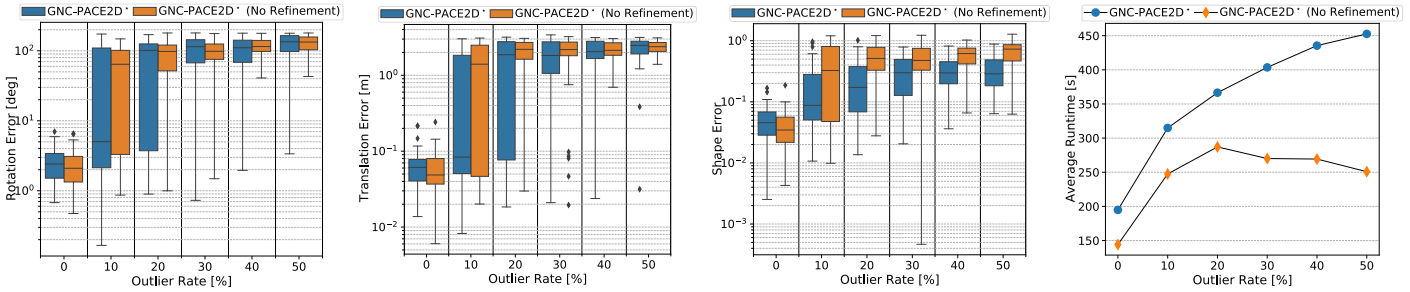


(c) Robustness of PACE3D# against increasing outliers on synthetic data: $N = 100$, $K = 50$, $r = 0.2$.

Fig. A5. Performance of PACE3D# compared to baselines in simulated experiments with different number of CAD models K and variation radius r. (a) The intra-class variation radius is increased to $r = 0.2$. (b) The number of CAD models is increased to $K = 50$. (c) $K = 50$ and $r = 0.2$. Each boxplot (and lineplot) reports statistics computed over 50 Monte Carlo runs.



(c) Robustness of PACE2D# against increasing outliers on synthetic data: $N = 10$, $K = 3$; $\boldsymbol{c}$ sampled from Δ_K uniformly at random.



(b) Performance of GNC-PACE2D* with and without local refinement.

Fig. A6. Additional results on PACE2D#. (a) Same as Fig. 8, but with shape parameters $\boldsymbol{c}$ sampled as a random one-hot vector. (b) Performance of GNC-PACE2D* with and without local refinement, see Appendix P.

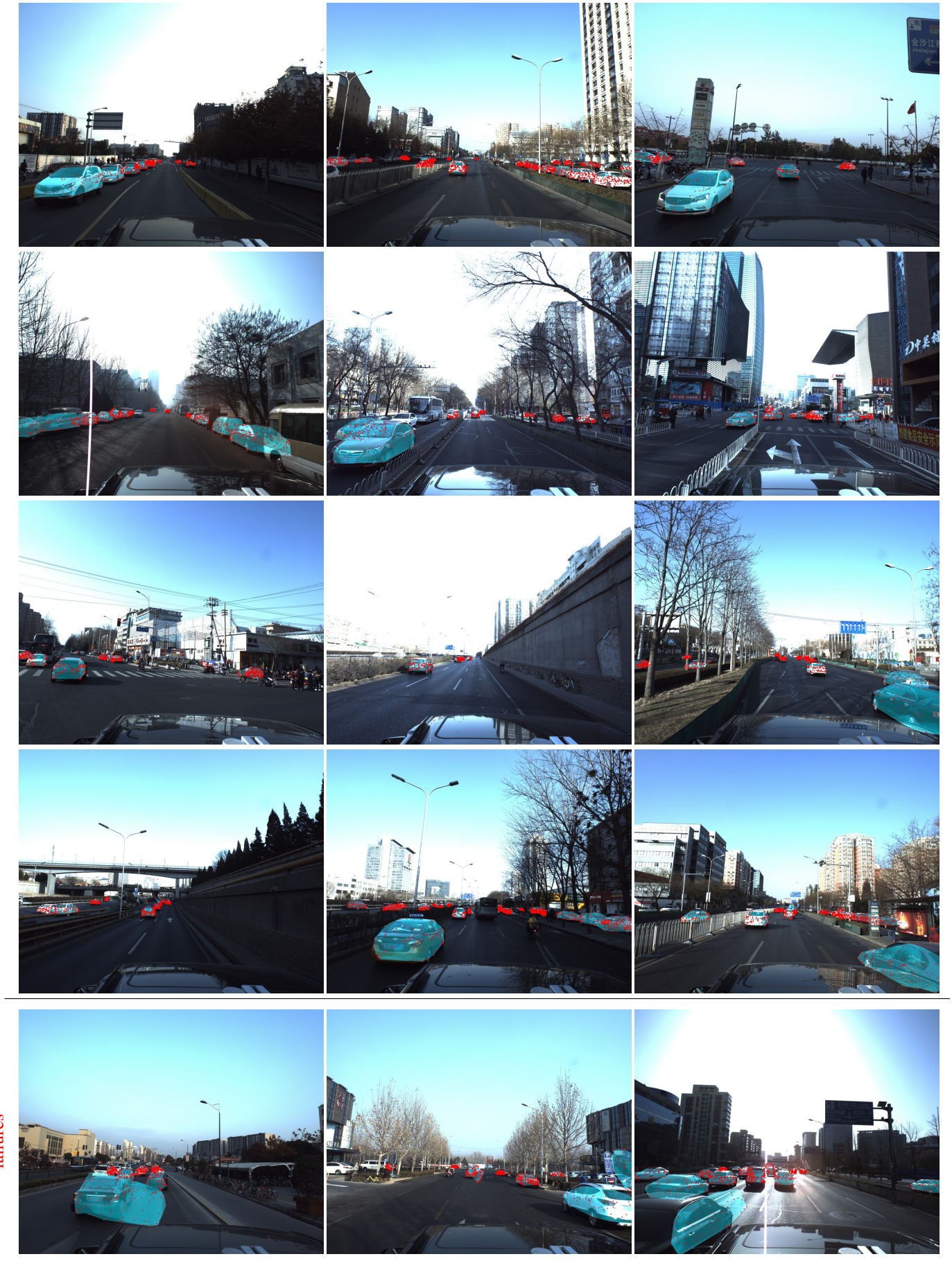


Fig. A6. Qualitative results for PACE3D#: overlay of estimated vehicle pose and shape on the images from the ApolloScape dataset. The images are manually selected out of the validation set in the dataset to showcase successful vehicle localization (top 4 rows) as well as failure cases (last row).

Fig. A6. Qualitative results for PACE2D#: overlay of estimated vehicle pose and shape on the images from the ApolloScape dataset. The images are manually selected out of the validation set in the dataset to showcase successful vehicle localization (top 4 rows) as well as failure cases (last row).

APPENDIX R
USING ROBIN TO UNCOVER MISLABELED GROUND-TRUTH ANNOTATIONS IN APOLLOSCAPE

This section shows that ROBIN is also able to detect incorrect ground-truth keypoint labels in the ApolloScape dataset. In particular, we tested ROBIN on the ground-truth keypoints provided by ApolloScape for training/validation/testing and manually inspected the points discarded as outliers by ROBIN. Intuitively, if the ground-truth annotations are correct (hence consistent with the CAD models), then all ground-truth keypoints must be included in the maximum hyperclique of the compatibility hypergraph, and ROBIN would detect zero outliers. On the other hand, if there are keypoints not included in the maximum hyperclique, then they might be potentially mislabeled keypoints.

We ran ROBIN on the ground-truth annotated keypoints in the validation set, which consists of 200 images with a total of 2674 car instances. For each instance in which ROBIN detected outliers (1566 counts), we then manually checked the keypoints against their semantic descriptions provided in [94]. We uncovered a total of 31 instances of incorrect ground-truth annotations, which is about 1.9% of the number of instances where ROBIN detects outliers.[7]

Image Name	Instance Index	Description
171206_034625454_Camera_5	3	#42 should be #38.
171206_034625454_Camera_5	11	#65 should be #22.
171206_035907482_Camera_5	3	#15 should be #19.
171206_065841148_Camera_6	0	#7 should be #50.
171206_070959313_Camera_5	0	#54 should be #8.
171206_071422746_Camera_5	4	#45 should be #41; #38 should be #37; #37 should be 36.
171206_074637632_Camera_5	3	#34 should be #26.
171206_080929510_Camera_5	5	#49 and #8 should be switched.
171206_082047084_Camera_5	0	#61 and #60 should be switched.
180114_023107908_Camera_5	4	#29 should be placed at the right exhaust below #31.
180114_025215740_Camera_5	5	#36 should be #21.
180114_025714151_Camera_5	2	#62, #63, #64, #65 should be #59, #58, #60, #61.
180114_030620413_Camera_5	18	#19 should be #24.
180116_031340200_Camera_5	18	#11 should be #9.
180116_032115845_Camera_5	4	#46 should be #33.
180116_040654119_Camera_5	6	#10 and #15 misplaced.
180116_041204649_Camera_5	0	#35 should be #22.
180116_041204649_Camera_5	5	#35 should be #22.
180116_053637003_Camera_5	4	#16 should be #41.
180116_053637003_Camera_5	10	#30 should be #24.
180116_053637003_Camera_5	11	#9 should be #48.
180116_053945714_Camera_5	6	#46 should be #11.
180116_064533064_Camera_5	17	#9 should be #12.
180116_065033600_Camera_5	11	#30 should be #33.
180118_070001729_Camera_5	11	Keypoints labeled are on two cars.
180118_070451170_Camera_5	5	#8 and #49 should be switched.
180118_070451170_Camera_5	13	#28 should be #33.
180118_071218867_Camera_5	5	#11 should be #29.
180118_071316772_Camera_5	4	#9 is mislabeled on another car.
180118_072006080_Camera_5	2	#42 should be #10.
180118_072006080_Camera_5	6	#11 should be #29.

TABLE A4
DETAILS ON ALL 31 MISLABELED INSTANCES WHERE ROBIN RETURNS NON-ZERO OUTLIERS.

[7]Since in this case we build the winding order dictionary via ray-tracing and due to the approximations induced by the discretization, ROBIN also returns false positives, where the ground-truth is correct while ROBIN detects outliers.

Image Name	Instance Index	Description
171206_072104589_Camera_5	8	#3 misplaced.
171206_074637632_Camera_5	5	#25 should be #35.
171206_080255599_Camera_5	1	#42 misplaced.
171206_080827230_Camera_5	3	#31 and #34 placed on another car.
180114_031156951_Camera_5	3	#9 should be #11.
180116_061243174_Camera_5	4	#1 should be #3.

TABLE A5
DETAILS ON 6 MISLABELED INSTANCES WHERE ROBIN RETURNS ZERO OUTLIERS.

Table A4 details them with the specific frames in which they are found. A common failure mode is the misplacement of keypoints between the two sides of the car. For example, placing #35 (defined as the top right corner of right rear car light in [94]) instead of #22 (defined as the top left corner of left rear car light). Fig. A8 and A9 show cropped images of the 31 mislabeled instances, with keypoints annotated. In particular, Fig. A8 shows an egregious case, with keypoint #9 being mislabeled as belonging to a completely different car on the other side of the frame (in this case, ROBIN successfully detects #9 as the sole outlier). We want to point out that due to the inaccuracies in the winding order dictionary, outliers determined by ROBIN might contain false positives. For example, in image 171206_071422746_Camera_5 (see Fig. A9), 6 keypoints are flagged by ROBIN as outliers, yet only 3 of them are actual outliers. Nevertheless, ROBIN still provides a viable way to validate the quality of 2D keypoints annotations based on 3D models.

A natural question to ask is how many mislabled keypoints are there in instances where ROBIN returns zero outliers. After all, it is possible for outliers to exist in degenerate configurations that pass ROBIN's compatibility checks consistently, making their ways into the maximum hyperclique. In addition, the dictionary of feasible winding orders obtained through ray-tracing is inherently only an approximation, due to the use of discretization. There are a total of 993 instances where ROBIN returns zero outliers. We manually checked all of them, and 6 of them contains mislabled keypoints, which is about 0.6%. This is 3 times lower than that with non-zero ROBIN outliers, suggesting that ROBIN is much more effective than a baseline method that randomly samples instances.

We want to stress that we are not trying to discredit the work behind ApolloScape. We sympathize with the authors and workers behind [94], as labeling 66 keypoints for all cars in the dataset is a herculean task. Our intent is to demonstrate that ROBIN can effectively detects outliers, and that it may also be used to validate human labels. In addition, exploring whether ROBIN can be used to enable self-supervision for detecting semantic keypoints will be an interesting topic for future research. For consistency with prior works, when calculating metrics in Section VIII, we used the original annotations, including mislabeled ground-truth keypoints.

Fig. A7. Mislabeled ground-truth keypoint annotations in ApolloScape detected by ROBIN. Green and red dots represent inliers and outliers determined by ROBIN, respectively. For a detailed description of the incorrect keypoint annotations for each image, we refer the reader to Table A4.

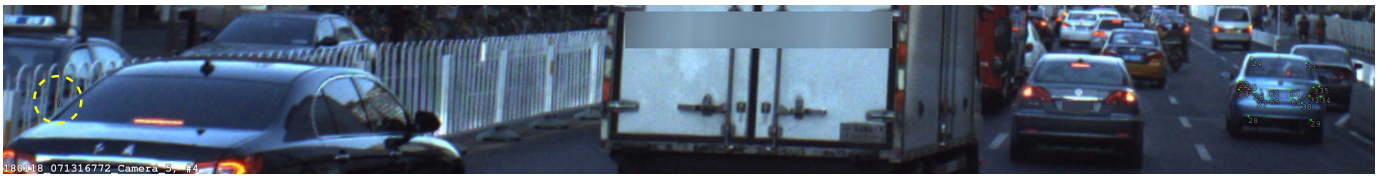


Fig. A8. Keypoint #9 (circled in yellow) is mislabeled in the 4th labeled instance in image 180118_071316772_Camera_5. Instead of being on the same car as the rest of the keypoints, #9 is misplaced as belonging to another car on the far left side.

Fig. A9. Mislabeled ground-truth keypoint annotations in ApolloScape that were not detected by ROBIN. Green dots represent inliers determined by ROBIN. For details, we refer the reader to Table A5.