2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

# Camera Pose Estimation from Bounding Boxes

Vaclav Vavra[1], Torsten Sattler[2], Zuzana Kukelova[1]

***Abstract*— Visual localization is an important part of many interesting applications, including robotics. The dominant localization strategy is to estimate the camera pose from 2D-3D matches between 2D pixel positions and 3D points. Yet, such approaches can be quite memory intensive and can lead to privacy risks. An interesting alternative to point-based matches is to use higher-level primitives for pose estimation. Consequently, this work investigates using correspondences between 2D and 3D bounding boxes for camera pose estimation. The resulting scene representation is compact and poses fewer privacy risks. In this setting, there are typically orders of magnitude fewer matches available compared to classical feature-based methods. In addition, the available correspondences are significantly more noisy. We investigate multiple strategies based on converting bounding box correspondences to point correspondences and propose a novel and simple 2-point camera absolute pose solver (DP2P) that exploits the fact that the depths of the objects can be approximated from the sizes of their bounding boxes.**

## I. Introduction

Camera pose estimation is the problem of computing the position and orientation from which a given image (or images) were taken. Camera pose estimation plays an important role in localizing robots in the world [26].

Arguably, the most popular approach to camera pose estimation, *e.g.*, used by feature-based visual localization systems [38], is based on 2D-3D correspondences between 2D image pixels and 3D points in the scene. The position and orientation of the camera are estimated by applying an absolute pose PnP solver, *e.g.*, the P3P solver for calibrated cameras [32], inside a RANSAC loop [23]. The required 2D-3D matches are typically obtained by matching the descriptors of local features extracted from the image with descriptors associated with 3D scene points.

3D point-based scene representations can easily become prohibitively expensive in terms of memory requirements, as (hundreds of) thousands or even millions of points are usually needed to describe a scene. An alternative to low-level primitives such as 3D points is to use higher-level primitives. In this work, we investigate using bounding boxes of objects for camera pose estimation. More precisely, we establish correspondences between 2D bounding boxes in an image and 3D bounding boxes in the scene, which in turn are used for pose estimation. We use 2D-3D point correspondences derived from the bounding box matches, *e.g.*, using the centers of the bounding boxes, as well as additional information about the dimensions of the bounding boxes for pose estimation. We chose bounding boxes for multiple reasons: (1) There are typically only relatively few objects in a scene (*e.g.*, around a hundred objects as opposed to tens of hundreds of thousands of 3D points), and each object can be described compactly using the bounding box and its center, as well as a class label, resulting in a highly memory efficient representation (*e.g.*, 25 bytes for an axis-aligned bounding box - 24 bytes for the two 3D points describing the box and 1 byte for the class label - as opposed to 140 byte for a 3D point position and its SIFT [25] descriptor). (2) Information about objects and their positions and orientation is typically necessary for robots operating in the real world, *e.g.*, to navigate to objects, manipulate them, or avoid obstacles. In contrast to point-based representations, the bounding box-based scene representation can thus also be used for tasks other than camera pose estimation. (3) Whereas image details can be recovered from points and their descriptors [33] (even when using smaller subsets of the points [6]), bounding boxes do not contain enough information to allow recovering images. Since it is possible to infer the presence and positions of objects from any robust localization approach [7], we thus consider bounding box-based representation as inherently more privacy-preserving.

Still, pose estimation from 2D-3D bounding box matches also comes with a set of challenges: (1) Bounding box detections can be quite imprecise. (2) In general, 2D bounding boxes are not projections of 3D bounding boxes and, therefore, only approximate point correspondences can be extracted from them.[1] (3) Compared to the large number of 3D points typically visible in an image, there are only a few bounding boxes visible in the image. This makes it harder to handle the higher noise in the measurements, *e.g.*, by averaging out errors during local optimization [23]. (4) Objects are often not distributed equally across the whole image. Coupled with the small number of potential matches, this means that degenerate configurations are much harder to avoid. *I.e.*, point correspondences extracted from bounding boxes can often be collinear in autonomous driving scenarios, which is a degenerate configuration for the P3P solver.

Still, bounding box-based pose estimation also offers advantages: (1) the sizes of the corresponding 2D and 3D bounding boxes can be used to approximate the depth of the object. This information can, in turn, be used to simplify pose estimation. (2) Many real-world objects are aligned with the

* This work was supported by the Czech Science Foundation (GACR) JUNIOR STAR Grant No. 22-23183M and the EU Horizon 2020 project RICAIP (grant agreement No. 857306).

[1]Visual Recognition Group, FEE, CTU in Prague
[2]Czech Institute of Informatics, Robotics and Cybernetics, CTU in Prague

[1]Machine learning can be used to predict the corners of 3D bounding boxes in images [35], [28], which in turn can be used to obtain additional 2D-3D point correspondences. These approaches need instance- or class-level training and are thus only applicable in scenes with known object types, limiting the generalizability of methods using them.

gravity direction. Thus, under some assumptions, 2D object-aligned bounding boxes can be used to approximate one or two rotation angles of a camera, decreasing the degrees of freedom (DoF) that have to be estimated from 6 to 5 or 4.

This paper investigates the use of bounding box correspondences for camera pose estimation. We make the following contributions: (1) We formulate different strategies to obtain information for camera pose estimation from bounding box matches, including strategies to obtain depth estimates from the dimensions of the boxes. (2) We derive a new solver (DP2P) that predicts the camera pose from two bounding box correspondences. This solver uses centers of bounding boxes and their depths approximated using the dimensions of the bounding boxes to estimate the 5DoF camera pose. Compared to the UP2P solver [20] that assumes a known gravity direction, the DP2P solver estimates one more angle of the rotation matrix, making it more robust to noise in gravity estimates and also applicable to situations where only one rotation angle is known. The proposed solver is general and can also be used with any depth information extracted, *e.g.*, from a RGB-D sensor. (3) We provide detailed experimental evaluations on both synthetic and real data to investigate the effectiveness of the different strategies and solvers, including detailed ablation studies. An important observation is that the P3P solver in many scenarios performs worse than the UP2P and DP2P solvers. This shows that the bounding box-based camera pose estimation problem differs from the classical point correspondence-based pose estimation problem, where the P3P solver is typically more robust and reliable than the UP2P solver. (4) We release code at github.com/vicsyl/pose-estimation-from-bounding-boxes.

## II. Related work

**Visual localization.** Localization approaches estimate the camera pose of a given query image in a known scene. State-of-the-art approaches rely on 2D-3D matches between 2D pixel positions in a query image and 3D points in the scene. These correspondences are either explicitly established using feature matching [38], [30], [26] or implicitly through regressing the corresponding 3D point per pixel [4]. The resulting 2D-3D matches are used for pose estimation by applying a PnP solver inside a RANSAC loop. While such approaches enable highly accurate camera pose estimation, with errors in the centimeter and sub-degree range, they can be quite memory intensive as they approximate the 3D scene structure using low-level primitives (3D points). We thus investigate using higher-level primitives, in the form of 3D bounding boxes, for pose estimation as they use less memory.

An alternative to visual localization via 2D-3D matches is to directly regress the camera pose from the query image [27], [41]. Such approaches, while efficient, currently are significantly (up to an order of magnitude) less accurate than methods based on 2D-3D matches [40]. Thus, we focus on camera pose estimation from 2D-3D matches.

**Object pose estimation.** Estimating the camera pose from 2D-3D bounding box correspondences, where each box corresponds to an object in the scene, is highly related to the object pose estimation problem. Object pose estimation algorithms aim to estimate the pose of a single object from a single image. These approaches are typically either trained per object instance [31] or class [24], [8] or require a 3D model of each object instance [42], [19] (or a similar-enough 3D model [21], [34]). We investigate a much simpler approach, which does not require 3D models or per-class training and instead only relies on bounding boxes.

**Object-based localization and mapping.** Objects have been used for both localization and 3D mapping, *e.g.*, for SLAM [37], [45], [3], [11], [17] or SfM [10], [16]. Most related to this work are methods that use bounding boxes for localization [43], [13], [15], [14], [45], [46] or mapping [10], [16], [29], [3], [11], [45]. Rather than directly using 2D and 3D bounding boxes, most works use 2D and 3D ellipses [10], [16], [29], [46], [13], [15], [14]. In 2D, they are obtained by fitting ellipses into 2D bounding box detections [46]. 3D ellipses are then obtained from 2D ellipses of the same object observed in two or more images [36]. Ellipses are preferred over bounding boxes as they lead to much simpler 3D fitting [36] and pose estimation [44], [46], [15], [13], [14] problems. Most related to this work is [46]. Their method uses one or more 2D-3D ellipse correspondences for camera pose estimation. They propose an approach to obtain 2D ellipses from 2D bounding boxes such that they are more consistent with the projections of the corresponding 3D ellipses. However, their approach requires training the fitting module for each observed instance of each observed object class. In contrast to ellipsis-based prior work, this paper focuses on directly using bounding boxes. To the best of our knowledge, there is no existing work that explores the potential strategies for bounding box-based camera pose estimation. Our work thus fills a gap in the literature.

## III. Pose estimation from bounding boxes

Below, we explore the information that 2D-3D bounding box correspondences can provide for pose estimation.

**Point correspondences.** A correspondence between a 2D bounding box detected in the image and a 3D bounding box can be transformed into 2D-3D point matches.

In general, a 2D bounding box is not a projection of a 3D bounding box, *i.e.*, the corners of the 2D bounding box do not need to correspond to any of the corners of the 3D bounding box. One reason is that even if the bounding boxes are object-aligned, for rotationally symmetric objects the 3D bounding box can be rotated arbitrarily around the rotational axis of the object. On the other hand, the bounding boxes are usually centered around the centers of objects. Due to the perspective projection, the center of the 2D bounding box is, in general, not in correspondence with the center of the 3D bounding box. Still, the centers of the bounding boxes usually provide a good approximation of a 2D-3D point correspondence [46] (see the approximate point correspondence $\mathbf{x}^C \leftrightarrow \mathbf{X}^C$ in Figure 1). These point correspondences can then be used for pose estimation, *e.g.*, in the classical P3P solver [32].

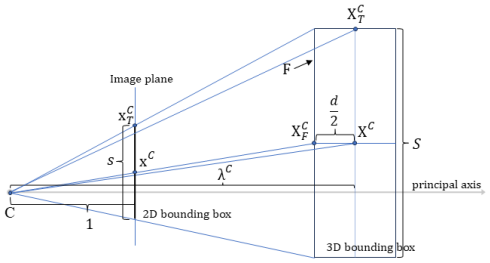


Fig. 1. Visualization of the assumptions under which the depth of the object can be estimated from the dimensions of the 2D and 3D bounding boxes based on similar triangles. See Sec. III.Dimensions/Depths for details.

Several other approximate 2D-3D point correspondences can be extracted from a 2D-3D bounding box correspondence. One such approximate 2D-3D match is between the middle point of the upper edge of the 2D bounding box and the center point of the upper face of the corresponding 3D bounding box (see the approximate point correspondence $\mathbf{x}_T^C \leftrightarrow \mathbf{X}_T^C$ in Figure 1). These points can be used either by a camera pose solver for pose estimation or for filtering out geometrically infeasible poses among several poses returned by the solver. These point correspondences are, in general, more noisy than the centers of the bounding boxes, but especially in the presence of a small number of bounding boxes from which some can be imprecisely detected, these additional correspondences can improve pose estimates.

**Dimensions/Depths.** Bounding boxes provide additional information in the form of their sizes (dimensions). The size of the 2D bounding box scales with the distance of the object from the camera. Thus, the dimensions of the 2D and 3D boxes provide (noisy) information about the object's depth.

Let us assume that the dimension of the 3D bounding box that is measured (*e.g.*, the height) is perpendicular to the principal axis of the camera, that the bounding boxes are object-aligned, and that the 3D bounding box is rotated so that one of its faces $F$ is parallel to the image plane (*c.f.* Fig. 1). In this case, assuming a calibrated camera, we can use triangle similarities to derive a simple formula for the depth $\lambda^C$ of the central point $\mathbf{X}^C$ of the 3D bounding box:

$$\lambda^C = \frac{S}{s} + \frac{d}{2} \ , \tag{1}$$

where $d$ is the depth of the point $\mathbf{X}^C$ inside the 3D bounding box measured in the direction of the principal axis of the camera, and $S$ and $s$ are sizes of the corresponding 3D and 2D bounding boxes, respectively.[2]

The assumptions under which Equation (1) holds are idealized and, in general, are not satisfied, *e.g.*, the 3D bounding box is usually not oriented such that one of its faces is parallel to the image plane. Moreover, we do not know the direction of the principal axis of the camera and therefore we cannot measure $d$ in the direction of the principal axis, and the dimension $S$ of the 3D bounding box so that it is perpendicular to the principal axis. However, several approximations can be made, so that the approximate depth can be computed from the information available for general bounding boxes: (1) $s$ and $S$ can be approximated using dimensions of 2D and 3D bounding boxes measured in the direction of their axes, *e.g.*, using the standard heights H and h of the bounding boxes. (2) $d$ in (1) may be equal to width $W$ or length $L$ of the bounding box, depending on its orientation, and thus $d$ can be approximated by their average $\frac{1}{2}(W+L)$. This means that we can use the approximation

$$\lambda^C \approx \frac{H}{h} + \frac{(W+L)}{4} \ . \tag{2}$$

Our experiments show that this approximation works well with real data for the combination of object-aligned 3D bounding boxes and axis-aligned 2D boxes.

To take advantage of this additional depth information, in Section IV, we propose a new minimal[3] solver for estimating the absolute pose of the camera. This DP2P solver uses two 2D-3D point correspondences and the known depths of these two 3D points to estimate a 5 DoF camera pose, *i.e.*, 2 DoF of rotation (assuming one known angle) and 3 DoF of translation. The solver even works with the depth of only one of the two 3D points, or the ratio of the two depths (see Section IV). The second approach is useful if the solver is used with depths that are known only up to an unknown common scale, *e.g.*, estimated using a neural network [12]. While in this paper we use (approximate) depth (2) obtained from bounding box dimensions, the proposed solver can be used with any source of depth, *e.g.*, depth maps from RGB-D sensors, with which many robots are equipped. When using depth information from such sensors, the pose estimates from the new DP2P solver will, in general, be even more precise.

**Gravity direction/Roll angle.** Many robots are equipped with an Inertial Measurement Units (IMU) that can be used to estimate two rotation angles of a camera/robot, *i.e.*, to align the vertical direction of a camera with the gravity direction. Even if such a sensor is not available, in many applications, the vertical direction of a camera mounted on a robot is by default aligned with the gravity direction. If such a robot moves on a ground plane, two rotation angles of the camera are fixed and known/zero. This allows us to use the 2-point absolute pose solver for the known vertical direction - the UP2P solver [20] - instead of the 3-point absolute pose solver (P3P) [32]. The point correspondences used by this solver can be extracted from the boxes as described above.

Sometimes the robot does not move on a ground plane and only one of the rotation angles is fixed (*e.g.*, when moving up the hill). This then leaves 5 DoF that need to be estimated. In such a situation, the new DP2P solver that assumes one known rotation angle can be applied.

Even if all three rotation angles of a camera are unknown, under some assumptions, bounding boxes can be directly

[2]Note, that under the mentioned assumptions $d = 2\left|\left|\mathbf{X}^C - \mathbf{X}_F^C\right|\right|$ and $\frac{S}{s}$ is the depth of the central point $\mathbf{X}_F^C$ of the face $F$, *i.e.*, the closest one among two faces parallel to the image.

[3]Minimal solvers are solvers that use the minimum number of input data, *e.g.*, point correspondences, to estimate all DoF by solving a system of equations with a finite number of solutions.

used to approximate one or two angles of a camera before running the solver. In general, many real-world objects are aligned with the gravity direction, *e.g.*, buildings, statues, poles, cars, furniture, *etc.*. Thus, from object-aligned 2D bounding boxes of at least two such objects, we can extract/approximate the vertical vanishing point. Subsequently, assuming that the camera is calibrated, we can align the camera vertical direction with the gravity direction.

Even one 2D object-aligned bounding box can, in some scenarios, provide information that can be used to simplify pose estimation. Assuming that the world coordinate system is aligned with the axes of one 3D bounding box, we can use the orientation of its corresponding 2D bounding box to approximate in-plane camera rotation, *i.e.*, camera roll angle.

## IV. DP2P MINIMAL SOLVER

Assuming a calibrated pinhole camera [18], the image projection $\mathbf{x}_i$ of a 3D point $\mathbf{X}_i$ can be written as

$$\lambda_i \mathbf{x}_i = \mathbf{R}(\mathbf{X}_i - \mathbf{C}) \ , \tag{3}$$

where $\mathbf{x}_i = (u, v, 1)^\top$ are the homogeneous coordinates of the image point, $\mathbf{R} = [r_{ij}]_{i,j=1}^3 \in SO(3)$ is a $3 \times 3$ rotation matrix, $\mathbf{C} \in \mathbb{R}^3$ is the camera center and $\lambda_i \in \mathbb{R}$ is the depth of the point $\mathbf{X}_i$ in the coordinate system of the camera.

Two 2D-3D point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{X}_i, \ i = 1, 2$, provide four constraint on the 6 DoF of $\mathbf{R}$ and $\mathbf{C}$. They restrict the position of the camera center $\mathbf{C}$ to lie on the surface of a torus (see Fig. 2 in [5]). In our new solver, in addition to two correspondences $\mathbf{x}_i \leftrightarrow \mathbf{X}_i, \ i = 1, 2$, we assume known depths $\lambda_i \ , i = 1, 2$. While we have two known depths, they provide only one constraint on the unknown camera pose. Assuming a calibrated camera, we can measure the projection angle $\theta = \angle(\mathbf{x}_1, \mathbf{x}_2) = \angle(\lambda_1 \mathbf{x}_1, \lambda_1 \mathbf{x}_2)$. Thus, from the known distance $\|\mathbf{X}_2 - \mathbf{X}_1\|$ and the depth $\lambda_1$, we can compute $\lambda_2$ or vice versa. The depth information restricts the solutions for $\mathbf{C}$ by one dimension to a circle on the surface of the torus, centered at the axis of revolution. This means that using two 2D-3D point correspondences and the known depths of the two 3D points (or their ratio), only a 5 DoF camera pose can be estimated.

In general, a solver that fixes any of the 6 DoFs of the camera pose can be derived. In practice, it usually makes sense to fix one of the rotation angles of the camera, *i.e.*, to assume that this angle is known. As described in Sect. III, in many robotics applications, it is reasonable to assume the roll angle of the camera to be zero (or known). Moreover the roll angle can, under some assumptions, be approximated from object-aligned 2D bounding boxes. Another practical situation is to assume that the pitch angle of the camera is zero (or known). This situation also appears in many practical scenarios. Here, let us assume that the roll angle of the camera is known. Using this angle, we can pre-rotate the camera such that its x-axis is oriented horizontally, *i.e.*, it lies in the $y = 0$ plane. This means:

$$\mathbf{R}^{-1}(1, 0, 0)^\top = \mathbf{R}^\top(1, 0, 0)^\top = (x, 0, z)^\top \tag{4}$$

and thus $r_{12} = 0$.

Let us first eliminate $\mathbf{C}$ from the projection equation (3) by considering the difference $\lambda_2 \mathbf{x}_2 - \lambda_1 \mathbf{x}_1$. This leads to

$$\lambda_2 \mathbf{x}_2 - \lambda_1 \mathbf{x}_1 = \mathbf{R}(\mathbf{X}_2 - \mathbf{X}_1) \ , \tag{5}$$

which gives us three linear equations in 8 unknown elements of the rotation matrix $\mathbf{R}$ (8 since $r_{12} = 0$).

The linear equation corresponding to the first row of (5), together with the quadratic constraint on the first row of $\mathbf{R}$ coming from the orthonormality of the rotation matrix $\mathbf{R}$, *i.e.*, the constraint $r_{11}^2 + r_{13}^2 = 1$, result in two equations in two unknowns. Note that in our case $r_{12} = 0$. This system has up to two solutions for the first row $\mathbf{r}_1 = (r_{11}, 0, r_{13})$ of $\mathbf{R}$. Plugging the solutions into the constraints on the second row of the rotation matrix $\mathbf{R}$, *i.e.*, the constraints $\mathbf{r}_1 \mathbf{r}_2^T = 0$ and $\mathbf{r}_2 \mathbf{r}_2^T = 1$, together with the equation corresponding to the second row of (5), gives us a system of two linear and one quadratic equations in the three unknowns of $\mathbf{r}_2$. This system has up to two solutions for the second row $\mathbf{r}_2 = (r_{21}, r_{22}, r_{23})$. The third row is simply $\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$. Excluding degenerate configurations and given a solution for $\mathbf{r}_1$, at most one of the solutions for $\mathbf{r}_2$ yields an $\mathbf{r}_3$ fulfilling the third row of (5). This means that, overall, there will only be up to two solutions to the original system (3) For a particular solution $\mathbf{R}$, we can get the solution for $\mathbf{C}$ as $\mathbf{C} = \mathbf{R}^{-1} \lambda_1 \mathbf{x}_1 - \mathbf{X}_1$.

A similar solver can be derived for a known pitch angle, *i.e.*, the horizontal z-axis. However, we do not need two separate solvers, since the same solver can be used in both cases after flipping the world coordinates $(x, y, z) \leftrightarrow (-z, y, x)$.

## V. EXPERIMENTAL RESULTS

The goal of our experiments is to study different strategies for camera localization from 2D-3D bounding box correspondences. To this end, we compare three different solvers: the standard P3P solver [32], the UP2P solver [20] that assumes a known gravity direction, and the new DP2P solver, which exploits information about known depths of 3D points. We investigate different strategies to run these solvers using information extracted from bounding box matches.

**Datasets.** We measured the performance on two datasets - Mapillary Metropolis [9], [1] and ARKitScenes [2]. The Mapillary Metropolis dataset contains images taken from a car in traffic in a flat area; therefore, the camera is very closely aligned with the vertical (gravity) direction, and most of the objects' 3D bounding boxes are nearly perfectly aligned with the vertical direction, with just small deviations. The 2D bounding boxes are image axis-aligned. [4] There are 8,002 images with at least two objects with both 2D and 3D bounding boxes. The distribution of the number of objects in the images in both datasets is shown in Tab. I.

Note that only $\sim 26\%$ respectively $\sim 3.5\%$ of the images in the Metropolis and ARKitScenes datasets contain more

[4]The 2D bounding boxes in this dataset are rectangular bounding boxes from equirectangular projection.

| # objects | ≥2 | ≥3 | ≥4 | ≥5 | ≥6 | 7+ |
|---|---|---|---|---|---|---|
| Metropolis | 8,002 | 4,957 | 3,195 | 2,106 | 1,326 | 1,983 |
| ARKitScenes | 113,929 | 50,956 | 12,688 | 4,015 | 1,135 | 305 |

TABLE I

STASTISTICS ABOUT SCENES WITH AT LEAST $n$ OBJECTS.

than 5 objects. This results in orders of magnitudes fewer matches compared to feature-based methods.

The ARKitScenes dataset contains indoor sequences, where the camera moves freely through the scene. This dataset represents a much more general camera motion than the motion usually present in robotics applications. However, we included this dataset in some experiments to show the performance of the tested approaches in such challenging scenarios. The 3D object bounding boxes in this dataset are, for the vast majority, aligned with the vertical direction (furniture standing on a horizontal floor). The dataset does not provide 2D bounding boxes, only per-pixel object instance labels. We used minimal area rectangles around these pixels to get the object-aligned 2D bounding boxes. Only those objects for which all vertices of their 3D bounding box project inside the image are considered.

Fig. 2 (left) shows the distribution of reprojection errors between the 2D bounding box centers and the projected 3D bounding box centers for both datasets. Fig. 2 (right) shows the accuracy of the depth estimates of the 3D bounding box centers measured as the ratio between the predicted depth obtained using Eq. (2) and the ground truth depth.

**Experimental setup.** Both datasets provide information about the objects that are visible in each image. We use this information to establish the 2D-3D bounding box matches: for each 2D box with the instance label $l$, we establish a match with a 3D box with the same label.[5] Rather than using random sampling (RANSAC), all potential minimal samples are considered for pose estimation. This is possible because there are only few objects visible in each image. Pose estimates are scored using a robust loss function [23] based on truncating the squared reprojection error at 12 pixels. Local optimization was not used, as we observed that it can decrease performance, likely due to the small number of objects in each scene, which do not provide enough information for pose refinement.

2D-3D bounding box matches provide us with point correspondences between their centers. These point correspondences are used for pose estimation in all tested solvers. For the DP2P solver, the 2D dimension of the object used in Eq. (2) as $h$ is computed as the distance between the midpoint of the upper and bottom edge of the bounding box.[6]

In addition, we explore three strategies for using extra correspondences from the top of the bounding boxes $\mathbf{x}_T^C \leftrightarrow \mathbf{X}_T^C$:

[5]In practice, one would establish correspondences between all bounding boxes with the same label. Given the small number of boxes per scene, the resulting set of matches can still be handled efficiently, especially if only two correspondences are needed for pose estimation.

[6]In this case, this height $h$ was selected as the dimension of the 2D bounding box that has the smallest angle from the vertical camera axis.

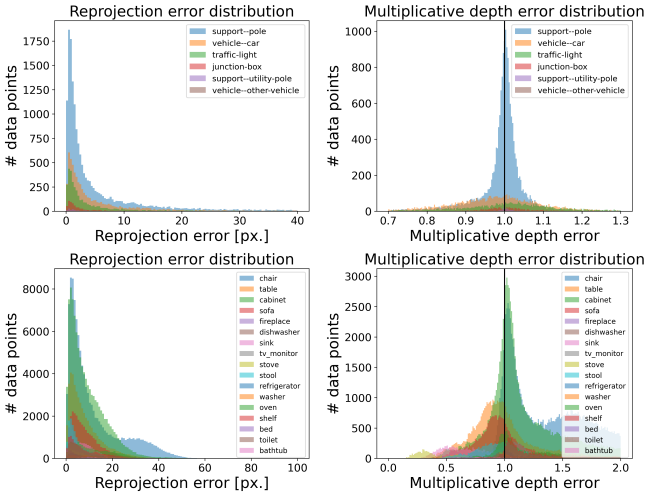


Fig. 2. **Dataset statistics.** Top - the Metropolis, bottom - the ARKitScenes dataset. Left - distribution of errors between the centers of the 2D bounding boxes and the projections of the centers of the corresponding 3D bounding boxes. Right - relative depth estimation error with the depths of the central points computed as $\lambda^C \approx \frac{H}{h} + \frac{(W+L)}{4}$, for the height of the 2D bounding box $h$, and the dimensions of the 3D bounding box $(H, W, L)$.

(1) The first strategy chooses a single model from multiple models returned by the minimal solver. To do this, each model is evaluated against the extra top correspondences of the minimal sample, and the best one is chosen. (2) The second strategy uses the extra correspondences for inlier counting, but not for pose estimation. (3) The third strategy uses the extra correspondences as part of minimal samples drawn in RANSAC. Figure 4 compares these strategies on the Metropolis dataset. The last strategy outperforms the others on this dataset. Thus, for all other experiments on Metropolis, we use the last strategy. For the ARKitScenes dataset the extra correspondences are much more noisy, and we thus do not use them in experiments on this dataset.

We use the P3P and UP2P solvers from PoseLib [22].

**Evaluation metrics.** For the real world data experiments we show the pose recall as a function of the position error (defined as the Euclidean distance between the estimated and ground truth camera center [39]) for a given threshold on the rotation error (defined as the angle of $\mathbf{R}_{gt}\mathbf{R}^T$) and vice versa. Pose recall is defined as the fraction of images with both position and orientation errors below given thresholds.

### A. *Controlled Experiments*

First, we analyze the robustness of the three solvers - UP2P, DP2P, P3P - to noise in the bounding box measurements and explore design choices for DP2P.

**Synthetic data.** In synthetic experiments, we measured the influence of (1) the reprojection errors between the 2D bounding box centers and the projected 3D bounding box centers, (2) the errors in depth estimates $\lambda^C$ (2), and (3) the deviations of the camera vertical direction from gravity, on the performance of the solvers. Due to space limits, we show here only the results for a combination of (2) and (3) with a fixed reprojection error of all points.

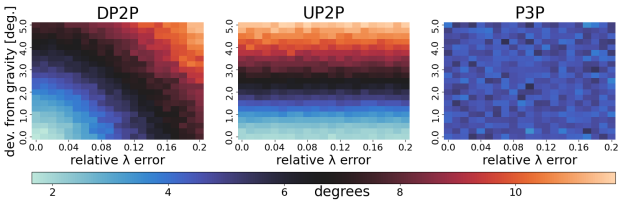


Fig. 3. **Distribution of mean rotation errors** depending on the deviation from gravity and relative depth ($\lambda$) error, for a fixed reprojection error of 0.01 in normalized image coordinates. DP2P uses a known pitch angle.

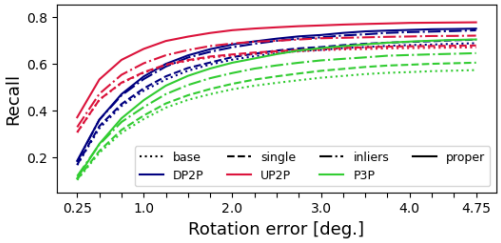


Fig. 4. **Using extra correspondences.** Comparison of different strategies for exploiting extra correspondences $\mathbf{x}_T^C \leftrightarrow \mathbf{X}_T^C$. Base - no extra correspondences are used. Single - a single model from multiple models returned by the minimal solver is chosen based on the inlier score on extra correspondences. Inliers - extra correspondences are used only for inlier counting. Proper - extra correspondences are treated as proper correspondences and are used also for estimation. Results are reported on the Metropolis dataset on images with at least 3 bounding boxes (4,957 images). The plot shows the recall for a position error threshold of 1 meter.

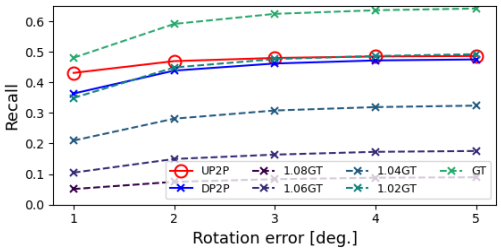


Fig. 5. **Depth sensitivity of the DP2P solver.** We compare the UP2P solver, the DP2P solver with known pitch and depth estimates from bounding boxes using Eq (2), and the DP2P solver with depths obtained by scaling all $\lambda_i^{gt}$ with a constant multiplicative factor *i.e.*, $\lambda_i = \lambda_i^{gt}(1+e_i)^{\pm 1}$, $e_i \in [0, 0.08]$. We show the fraction of images localized within 0.5 meters of the ground truth pose as a function of the rotation error.

For each synthetic scene instance, the ground truth normalized image coordinates $\mathbf{x}_i^{gt}$ were sampled uniformly from $[-1,1]^2$, the ground truth depths $\lambda_i^{gt}$ uniformly from $[2,75]$, and the rotation matrix $\mathbf{R}$ was sampled uniformly from rotations with the given deviation from gravity. The 3D bounding box centers $\mathbf{X}_i^C$ were obtained by backprojecting $\mathbf{x}_i^{gt}$ using given $\mathbf{R}$ and $\lambda_i^{gt}$. Finally, the depth estimates $\lambda_i$ used by the DP2P solver were computed by scaling $\lambda_i^{gt}$ with a multiplicative factor, *i.e.*, $\lambda_i = \lambda_i^{gt}(1+e_i)^j$, where $e_i, j$ were sampled uniformly from $[0, 0.2]$ and $\{-1, 1\}$, respectively. The 2D bounding box centers $\mathbf{x}_i$ were computed from $\mathbf{x}_i^{gt}$ by adding a random offset with the norm of 0.01 to $\mathbf{x}_i^{gt}$ in a random direction (corresponding to an error of 5 pixels for a focal length of 1,000 pixels and a field of view of 60 degrees). We generated 2,000 scene samples per given set of noise parameters. Fig. 3 shows the mean rotation errors returned by the solvers for a given relative depth error (in the heatmap bin $[a, a+0.01]$ contains depth noises $\frac{e_1+e_2}{2} \in [a, a+0.1]$), and camera deviation from gravity. We compared DP2P with known pitch, UP2P, and P3P. By their definitions, P3P is not affected by errors in the gravity estimates and relative depth estimates, and UP2P is only affected by errors in the gravity estimates. For the chosen reprojection error (0.01 in normalized coordinates), DP2P performs equally well or better than UP2P for the relative depth error below $8\%$. Moreover, it performs better than P3P for relative depth errors below $8\%$ and deviations from gravity below 2 deg. UP2P outperforms P3P for deviations from gravity below 1 deg. For Metropolis, most of the depth estimates obtained using Eq. (2) had errors below $10\%$ (*c.f.* Fig. 2).

**Strategies for obtaining depth estimates for the DP2P solver.** Eq. (2) provides initial depth estimates $\{\lambda_i^e\}_{i=1}^2$ for the two objects used by the DP2P solver. As they are approximate, they do not satisfy the cosine law given by the known projection angle $\theta = \angle(\mathbf{x}_1, \mathbf{x}_2)$ and the side lengths $\|\lambda_1^e \mathbf{x}_1\|$, $\|\lambda_2^e \mathbf{x}_2\|$ and $\|\mathbf{X}_2^C - \mathbf{X}_1^C\|$. To compensate for noise in the estimated $\{\lambda_i^e\}_{i=1}^2$, we can use $\theta$ and $\|\mathbf{X}_2^C - \mathbf{X}_1^C\|$ to compute new values $\{\lambda_i^C\}_{i=1}^2$ that satisfy the cosine law. These new values are then used as input to the DP2P solver.

There are several strategies on how this can be done. (1) We can fix the ratio $r = \lambda_2^e / \lambda_1^e$ and calculate $\{\lambda_i^C\}_{i=1}^2$ so that they satisfy the cosine law and $r = \lambda_2^C / \lambda_1^C$. (2) We can fix one of the estimated depths, *e.g.*, the first, set $\lambda_1^C = \lambda_1^e$ and compute $\lambda_2^C$ from $\lambda_1^C$, $\theta$ and $\|\mathbf{X}_2^C - \mathbf{X}_1^C\|$. This leads to a quadratic equation with possibly two positive roots. Both roots can be used within DP2P. Yet, picking the root that is closer to the initial estimate $\lambda^e$ was shown to be sufficient.

One estimated depth can be significantly more noisy than the second, and it is generally not known which one is more accurate. Thus, it is important to test both variants *i.e.*, fix $\lambda_1^e$ as well $\lambda_2^e$ and run the DP2P solver twice. This can result in a significant improvement in performance. In general, one can run the solver for each minimal sample three times and also evaluate (inside RANSAC) the variant with fixed ratio. However, including estimates from the ratio $r$ did not bring any improvement on the tested datasets. Thus, we did not use this strategy in our experiments. Note that using the ratio strategy can be very useful in scenarios where depths will be estimated using neural networks. In such cases, the returned depths are usually known only up to an unknown common scale. Taking the ratio eliminates this unknown scale.

**Depth sensitivity of the DP2P solver.** The performance of the DP2P solver depends directly on the quality of the depth estimates. To measure this dependency under realistic conditions, on the Metropolis dataset, we run the DP2P solver with depth estimates computed by scaling all $\lambda_i^{gt}$ with a constant multiplicative factor *i.e.*, $\lambda_i = \lambda_i^{gt}(1+e_i)^{\pm 1}$, $e_i \in [0, 0.08]$. Fig. 5 shows the results for different noise levels as well as the depth estimates computed from the bounding boxes using Eq. (2) (solid blue) and the results

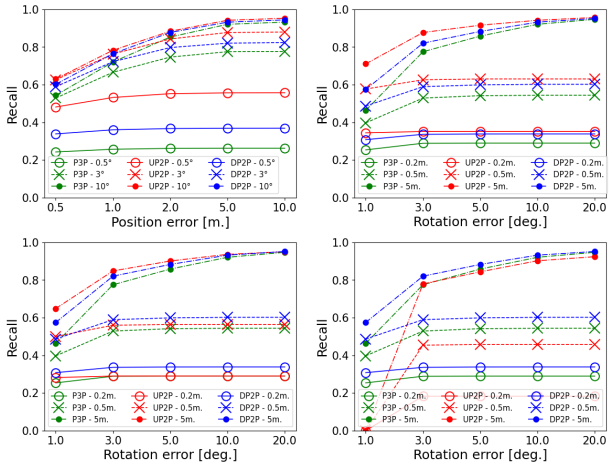


Fig. 6. **Comparison of DP2P (with known pitch), UP2P, and P3P on images from the Metropolis dataset** observing at least three objects. In the general comparison for rotation (top left) and position error levels (top right), the performance of DP2P is better than P3P but below UP2P. UP2P is quite sensitive to deviations from the gravity direction, which was simulated by rotating the image around the principle axis. For a small rotation by $0.4°$ (bottom left), the performance of UP2P already drops. For a rotation by $1°$ (bottom right) the performance of UP2P drops significantly.

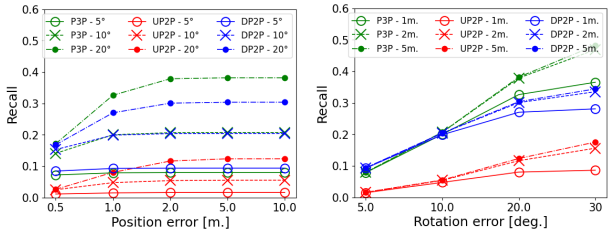


Fig. 7. **Comparison of DP2P (with known roll), UP2P, and P3P on all images from the ARKitScenes dataset** observing at least three objects. While P3P performs the best on generic poses of the dataset, DP2P performs not much worse, especially on tighter error levels. As expected, the performance of UP2P is much worse. Measured on 50,956 images.

of the UP2P solver (solid red). The results for DP2P with depths estimated from the bounding boxes are quite good and correspond to the results of DP2P with a constant $2\%$ error in depths. This is thanks to the fact that for the DP2P solver, it is sufficient to have one precise depth (*c.f.* strategies for obtaining depth estimates). Still, there is a gap between the results for ground truth depths and estimated depths. This means that more precise depth estimates obtained, *e.g.*, using a depth sensor or learning-based method, can improve the performance of the DP2P solver.

### *B. Comparison on Real World Data*

Comparisons of the three minimal solvers - DP2P, UP2P, and P3P - on images from the Metropolis and ARKitScenes datasets with at least visible 3 bounding boxes are shown in the Fig. 6 (top row) respectively Fig. 7.

On the Metropolis dataset, where the camera's vertical directions are almost perfectly aligned with the gravity direction, the best-performing solver is UP2P. On the ARKitScenes dataset with more general camera motion, the best-performing solver is P3P that estimates full 6DoF motion.

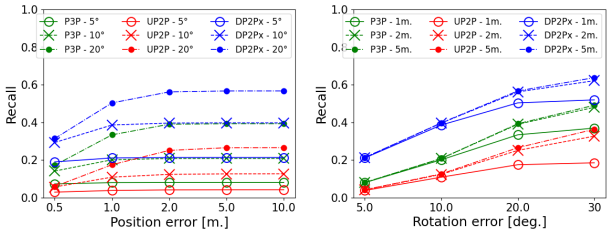


Fig. 8. **Comparison of DP2P (with known roll), UP2P, and P3P on images from the ARKitScenes dataset** observing at least three objects; for images with deviations within 2 degrees of the camera roll. Measured on 16,558 images. The best-performing solver is DP2P.

The performance of the DP2P solver is somewhere between P3P and UP2P showing good performance on both datasets.

While the DP2P solver estimates two rotation angles, the UP2P solver estimates only one (yaw). Therefore, UP2P is more sensitive to noise in the gravity direction. This can be observed from Fig. 8, showing results on 16,558 images from the ARKitScene dataset where the ground truth camera roll was within $2°$. It can be seen that for such images, the DP2P solver with known/zero roll significantly outperforms both UP2P and P3P. Similar behavior can be observed from Fig. 6, where we simulated noise in the given gravity estimate by rotating images from the Metropolis dataset around the principal axis. While this favors the DP2P solver that estimates this angle, it simulates a situation that is common in practice. As shown in Fig. 6 (bottom row), already for a small rotation by $0.4°$, DP2P outperforms UP2P, which is quite sensitive to this type of noise, on finer thresholds (0.2m, 0.5m) on the position errors of the estimated poses. Although P3P is robust to this rotation and estimates all rotation angles, it again performs significantly worse than DP2P. This is an interesting and important observation, as for classical point correspondences, the P3P solver typically provides more accurate estimates than the solvers that are fixing some rotation angles (as P3P is not affected by noise in these angles). We attribute this to the small number of correspondences, and the higher noise level in the correspondences. Another reason might be that often, especially for Metropolis, all three 3D box centers are nearly collinear, which is a degenerate configuration for P3P.

Note that all solvers perform better on Metropolis than on ARKitScenes, most likely due to the higher reprojection errors for the latter (see Fig. 2).

## VI. Conclusions

In this paper, we have investigated the problem of camera pose estimation from correspondences between 2D and 3D bounding boxes. This problem brings several advantages over classical feature-based localization methods. First, the bounding box-based scene representations are compact. Second, these representations are very abstract, and thus do not contain private information of users. On the other hand, using bounding boxes comes with new challenges, *e.g.*, orders of magnitude fewer matches, and significantly more noisy measurements. Therefore, we tested the performance of

different camera pose estimation solvers and investigated different strategies for running these solvers using information extracted from bounding box matches. To exploit additional information available from bounding boxes, we proposed a novel minimal solver (DP2P) for pose estimation using two point correspondences and known depths. This solver is general and can be used with any source of depth estimates. We show that when using depth estimates of objects obtained from the sizes of their bounding boxes, the DP2P solver performs quite well over different camera motions/poses. For poses with one rotation angle small, *e.g.*, negligible roll, it significantly outperforms the P3P and UP2P solvers.

As expected, and as evident from our results, using bounding boxes leads to rather coarse camera pose estimates that are significantly less accurate than feature-based methods. Still, given their small memory footprint, bounding box-based methods are well-suited for coarse localization, *e.g.*, as an intermediate step for feature-based methods or to initialize a refinement step that better aligns the image with 3D models of the objects inside the bounding boxes.

## REFERENCES


[1] Mapillary Metropolis Dataset. https://www.mapillary.com/dataset/metropolis, 2021. 4

[2] G. Baruch, Z. Chen, A. Dehghan, T. Dimry, Y. Feigin, P. Fu, T. Gebauer, B. Joffe, D. Kurz, A. Schwartz, and E. Shulman. Arkitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *NeurIPS*, 2021. 4

[3] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic data association for semantic slam. In *ICRA*, 2017. 2

[4] E. Brachmann, T. Cavallari, and V. A. Prisacariu. Accelerated coordinate encoding: Learning to relocalize in minutes using rgb and poses. In *CVPR*, 2023. 2

[5] F. Camposeco, T. Sattler, A. Cohen, A. Geiger, and M. Pollefeys. Toroidal Constraints for Two-Point Localization under High Outlier Ratios. In *CVPR*, 2017. 4

[6] K. Chelani, F. Kahl, and T. Sattler. How privacy-preserving are line clouds? recovering scene details from 3d lines. In *CVPR*, 2021. 1

[7] K. Chelani, T. Sattler, F. Kahl, and Z. Kukelova. Privacy-preserving representations are not enough: Recovering scene content from camera poses. In *CVPR*, 2023. 1

[8] X. Chen, Z. Dong, J. Song, A. Geiger, and O. Hilliges. Category level object pose estimation via neural analysis-by-synthesis. In *ECCV*, 2020. 2

[9] A. Colovic, A. Knapitsch, L. Porzi, S. R. Bulò, J. Mislej, M. Lopez-Antequera, V. Balntas, E. Miller, Y. Kuang, and P. Kontschieder. ICCV 2021 Tutorial "Benchmarking City-Scale Semantic 3D Map Making with Mapillary Metropolis". https://research.mapillary.com/MetropolisTutorial, 2021. 4

[10] M. Crocco, C. Rubino, and A. Del Bue. Structure from motion with objects. In *CVPR*, 2016. 2

[11] J. Dong, X. Fei, and S. Soatto. Visual-inertial-semantic scene representation for 3d object detection. In *CVPR*, 2017. 2

[12] A. Eftekhar, A. Sax, J. Malik, and A. Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *ICCV*, 2021. 3

[13] V. Gaudilliere, G. Simon, and M. Berger. Camera relocalization with ellipsoidal abstraction of objects. In *ISMAC*, 2019. 2

[14] V. Gaudillière, G. Simon, and M. Berger. Perspective-1-ellipsoid: Formulation, analysis and solutions of the camera pose estimation problem from one ellipse-ellipsoid correspondence. *IJCV*, 131(9):2446–2470, 2023. 2

[15] V. Gaudillière, G. Simon, and M.-O. Berger. Perspective-2-ellipsoid: Bridging the gap between object detections and 6-dof camera pose. *IEEE Robotics and Automation Letters*, 5(4):5189–5196, 2020. 2

[16] P. Gay, C. Rubino, V. Bansal, and A. Del Bue. Probabilistic structure from motion with objects (psfmo). In *ICCV*, 2017. 2

[17] D. Gálvez-López, M. Salas, J. D. Tardós, and J. Montiel. Real-time monocular object slam. *Robotics and Autonomous Systems*, 75:435–449, 2016. 2

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2nd edition, 2004. 4

[19] X. He, J. Sun, Y. Wang, D. Huang, H. Bao, and X. Zhou. Onepose++: Keypoint-free one-shot object pose estimation without CAD models. In *NeurIPS*, 2022. 2

[20] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-Form Solutions to Minimal Absolute Pose Problems with Known Vertical Direction. In *ACCV*, 2010. 2, 3, 4

[21] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic. Megapose: 6d pose estimation of novel objects via render & compare. In *CoRL*, 2022. 2

[22] V. Larsson. PoseLib - Minimal Solvers for Camera Pose Estimation, 2020. 5

[23] K. Lebeda, J. Matas, and O. Chum. Fixing the Locally Optimized RANSAC. In *BMVC*, 2012. 1, 5

[24] Y. Lin, J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield. Single-stage keypoint-based category-level object pose estimation from an RGB image. In *ICRA*, 2022. 2

[25] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004. 1

[26] S. Lynen, B. Zeisl, D. Aiger, M. Bosse, J. Hesch, M. Pollefeys, R. Siegwart, and T. Sattler. Large-scale, real-time visual–inertial localization revisited. *IJRR*, 39(9):1061–1084, 2020. 1, 2

[27] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle. LENS: Localization enhanced by NeRF synthesis. In *CoRL*, 2021. 2

[28] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 1

[29] L. Nicholson, M. Milford, and N. Sunderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, PP:1–1, 08 2018. 2

[30] V. Panek, Z. Kukelova, and T. Sattler. Meshloc: Mesh-based visual localization. In *ECCV*, 2022. 2

[31] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019. 2

[32] M. Persson and K. Nordberg. Lambda twist: An accurate fast robust perspective three point (p3p) solver. In *ECCV*, 2018. 1, 2, 3, 4

[33] F. Pittaluga, S. J. Koppal, S. B. Kang, and S. N. Sinha. Revealing Scenes by Inverting Structure From Motion Reconstructions. In *CVPR*, 2019. 1

[34] G. Ponimatkin, Y. Labbé, B. Russell, M. Aubry, and J. Sivic. Focal length and object pose estimation via render and compare. In *CVPR*, 2022. 2

[35] M. Rad and V. Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *CVPR*, 2017. 1

[36] C. Rubino, M. Crocco, and A. D. Bue. 3d object localisation from multi-view image detections. *IEEE TPAMI*, 2018. 2

[37] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *CVPR*, 2013. 2

[38] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From Coarse to Fine: Robust Hierarchical Localization at Large Scale. In *CVPR*, 2019. 1, 2

[39] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. Benchmarking 6DOF Urban Visual Localization in Changing Conditions. In *CVPR*, 2018. 5

[40] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixé. Understanding the limitations of cnn-based absolute camera pose regression. In *CVPR*, 2019. 2

[41] Y. Shavit, R. Ferens, and Y. Keller. Learning Multi-Scene Absolute Pose Regression With Transformers. In *ICCV*, 2021. 2

[42] J. Sun, Z. Wang, S. Zhang, X. He, H. Zhao, G. Zhang, and X. Zhou. OnePose: One-shot object pose estimation without CAD models. *CVPR*, 2022. 2

[43] M. Toso, M. Taiana, S. James, and A. Del Bue. You are here! finding position and orientation on a 2d map from a single image: The flatlandia localization problem and dataset. In *arXiv:2304.06373*, 2023. 2

[44] M. Zins, G. Simon, and M. Berger. 3d-aware ellipse prediction for object-based camera pose estimation. *3DV*, 2020. 2

[45] M. Zins, G. Simon, and M. Berger. Oa-slam: Leveraging objects for camera relocalization in visual slam. In *ISMAR*, 2022. 2

[46] M. Zins, G. Simon, and M.-O. Berger. Object-Based Visual Camera Pose Estimation From Ellipsoidal Model and 3D-Aware Ellipse Prediction. *IJCV*, 130(4):1107–1126, apr 2022. 2