Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image - Supplementary Material -

Eric Brachmann*, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, Carsten Rother
TU Dresden
Dresden, Germany
*eric.brachmann@tu-dresden.de

This supplementary material provides useful details for re-implementing our approach. We give a formal definition of our 2D projection measure and a complete listing of parameter settings used in our experiments. Then, we expand on qualitative and quantitative results. Finally, we explain how the supplementary video was created. In summary, this document contains:

- Implementation details
- Details on the 2D projection measure
- List of parameter settings
- Additional results
- Details on the supplementary video

1. Implementation Details

1.1. Standard Random Forest

This subsection contains details on the training of stack level 0. We will expand on this with regard to object coordinate auto-context in the next subsection.

We draw training samples uniformly from within the object segmentation and up to a certain distance outside the segmentation. This distance is set to 50% of the maximum feature size. We additionally draw samples of the background class from a selection of random interior background images.

To be scale invariant during test time, the forest has to be trained with different sizes of training images. We approximate re-scaling training images by scaling the feature offsets instead. The scale factor is chosen randomly per training sample. The scale range depends on the minimal and maximal object scale expected at test time. We use a similar approach when adding in-plane rotation to training images. We do not actually rotate training images but we rotate the feature offset vectors instead.

At each tree depth level, a pool of features is created by sampling feature parameters uniformly. Feature thresholds are also chosen randomly by calculating the feature response at a random training pixel. Should a feature access a pixel outside the object segmentation during training we return uniform color noise.

When learning the empirical distribution in the forest leafs, we increase the number of training samples passed through the tree by a factor of 3. We keep at most 2000 samples per leaf and object before running mean-shift. Modes with less than 10 samples supporting it are dismissed. We calculate full co-variance for each mode, setting the mean of the distribution to the mode coordinate.

1.2. Object Coordinate Auto-Context Forest

Starting with the second layer of the stack, we calculate the prediction of the previous forest for each training image. Since our training images are segmented and contain no background, we paste each training image into a random image of our background image set at a random position. The resulting montage is most likely physically implausible, but we found this simple strategy sufficient. We calculate the forest prediction on the montage. The resulting auto-context feature channels are stored sub-sampled. This reduces the memory footprint, and increases the effective range of smoothing operations between auto-context layers.

Before training the next layer, we also calculate object probabilities of the previous layer on the background image set. The background images represent our set of negative samples, hence object probabilities should be low. Background regions with high object probability represent hard negatives, *i.e.* samples where the prediction of the last layer was wrong. When training the new layer, we draw 50% of the background training samples according to object probability. Thus, sampling has a bias towards hard negatives. We found this giving a slight but no substantial gain in segmentation performance of the forest.

1.3. RANSAC

We draw an hypothesis by sampling 4 pixels according to $P_i^D(c)$. Pixels 2-4 are sampled in the vicinity of the first pixel chosen. We use the following heuristic to determine a sensible search radius: First, we read out the object coordinate prediction $\mathbf{y}^1$ associated with the first pixel chosen. We calculate the maximum distance of $\mathbf{y}^1$ to the 3D object bounding box. We project this distance into the image, assuming a worst case depth of 30cm (minimal object distance from the camera). The result is a worst case search window depended on the object size. We relax the worst case scenario, by shrinking the search window to 30% size.

We reject configurations of 4 pixels which do not pass the following tests: The minimum distance of all pixels in image space should be at least 10px. The minimum distance of all pixels in object space should be at least 10mm. None 3 pixels should be co-linear in object space. Hence, we enforce a minimum distance of 10mm between the line formed by two pixels and the third pixel. Finally, the reprojection error of the 4 correspondences should be below the inlier threshold. If after 1M iterations no valid pixel set has been found, we abort. We discard a hypothesis if the resulting 2D bounding box occupies less than 400 pixels.

We use two different implementations of perspective-n-point. We use the approach of [3] when calculating a hypothesis from 4 correspondences (it gave the most stable results in the minimal setting). We use the approach of [8] to re-calculate a hypothesis based on the complete inlier set (it was stable in the general setting and fast). We use the implementations available in OpenCV [2].

When re-calculating the hypothesis based on inlier correspondences we use at most 1000, randomly chosen with replacement. For full refinement using uncertainty, we run 100 iterations of [9] (gradient free). We use the implementation of NLopt [6]. When calculating the log-likelihood of each pixel, we ignore the contribution of mixture components whose covariance matrix has a determinant of less than 1000 (object coordinates are measured in mm). We threshold the log-likelihood of each pixel between -100 and 100 for robustness.

2. 2D Projection Measure

This measure is calculated using a 3D model or a point cloud of the object. We accept an estimated pose, if the average re-projection error of all object model vertices is below 5px. To calculate the re-projection error, we project the vertices into the image using the ground truth pose and the estimated pose. We calculate the average distance of the projections of corresponding vertices:

$$\frac{1}{|\mathcal{V}|} \sum_{\mathbf{v} \in \mathcal{V}} ||CH_c\mathbf{v} - C\tilde{H}_c\mathbf{v}||_2 < 5px, \tag{1}$$

where $\mathcal{V}$ is the set of all object model vertices, C is the camera matrix, H_c is the ground truth pose and $\tilde{H}_c$ is the estimated pose. We assume normalization of the homogeneous vector before calculating the L_2 norm.

3. List of Parameter Settings

In this section, we report the parameter settings used in our experiments. First, we report our default parameter set. Then, we report deviating parameters for specific experiments, *e.g.* for the RGB-D variant of our system or for camera localization. For the binaries of Brachmann *et al.* we used the default settings reported in [1]. For the LINE2D baseline [5] we used the standard settings of OpenCV [2]. For the color check post-processing we use the thresholds reported in [4].

3.1. Standard Setting

Trees in the forest: 3
Stack level: 3
Samples drawn per object: 500k
Samples drawn from background: 1.5M
Features sampled: 1000
Maximum feature offset: 10px
Object coordinate proxy classes: 125
Maximal tree depth: 64
Minimal number of samples in leafs: 50
Bandwidth of mean-shift: 25mm
Neighborhood used in $g^d_{\mathcal{C}}$: 5×5
Sub-sampling of auto-context feature channels: 2
Neighborhood used in $\mathbf{g}^d_{\mathcal{Y}}$: 3×3
Inlier threshold: 3px
Hypotheses sampled: 256
Pixel batch size: 100000
Full refinement iterations: 100
Full refinement bounds X/Y/Z/Rot:
$\pm 50mm/50mm/200mm/10°$

3.2. Object Pose Estimation (RGB-D)

Parameters used in the RGB-D variant of our system for estimating object poses (see Sec. 4.1.2).

Maximum feature offset: 10px $\times$ m (depth normalized)
Inlier threshold: 10mm

3.3. Camera Localization (RGB)

Parameters used in the RGB variant of our system for camera localization (see Sec. 4.3).

Samples drawn per scene: 2.5M
Maximum feature offset: 20px
Inlier threshold: 10px
In-plane rotation range (training): -30° to +30°
Sub-sampling of auto-context feature channels: 4

To achieve equal run-time of the two RANSAC variants *w/ sharing* and *w/o sharing* we adjusted parameters in the following way: The final hypothesis of the variant *w/ sharing* will at least pass 8 rounds of pre-emptive RANSAC (drawing of pixel batches and re-solving PnP), then it is refined using uncertainty. For the variant *w/o sharing* there is no minimum number of pre-emptive passes. This results in less pixels drawn and less iterations of re-solving PnP. Also, there is no refinement using uncertainty. Using these settings results in a run-time of ca. 700ms for both variants to process all 7 scenes per test image.

3.4. Camera Localization (RGB-D)

Parameters used in the RGB-D variant of our system for camera localization (see Sec. 4.3).

Samples drawn per scene: 2.5M
Maximum feature offset: 50px $\times$ m (scale normalized)
Inlier threshold: 100mm
In-plane rotation range (training): -30° to +30°
Sub-sampling of auto-context feature channels: 4

4. Results

For the data set of Hinterstoisser *et al.* [4] we report rates of correctly estimated poses per object in Table 1. We also list median pose errors in x/y/z and rotation. Qualitative results for individual objects are shown in Fig 1, and for multiple objects in Fig 2. These figures also show the intermediate output of the auto-context random forest at different stack levels.

We show results on individual scenes of the data set of Shotton *et al.* [10] in Table 2. Qualitative results are shown in Fig 3.

5. Supplementary video

We created a video showing different visual effects to demonstrate the accuracy of our pose estimation system. The system consists of three different sequences. The first two sequences show results for object pose estimation on the data set of Hinterstoisser *et al.* [4], the third sequence shows results for camera localization on the data set of Shotton *et al.* [10].

5.1. Object Pose Estimation Sequences

The data set of Hinterstoisser *et al.* [4] contains 1000+ frames per object. However, these sequences are assemblies of separate shots of the scene, *i.e.* there are no smooth camera transitions. To create a pseudo-smooth image sequence we selected frames in the following way:

We transformed the camera pose of each frame to azimuth/elevation/distance. We clustered all frames according to 5° azimuth bins. For each bin, we aim at selecting one image resulting in a sequence as smooth as possible. Therefore, we formulate a chain-MRF with 72 nodes representing 72 frames, and a pairwise term which ensures that two neighboring frames:

- belong to neighboring azimuthal bins,
- have close 2D projections of object centers (euclidean distance),
- have similar distances of the camera to the object (euclidean distance),
- have a small angular distance in camera pose.

We solve this formulation to create one image sequence per object. Then, we apply our pose estimation method to the sequence, and add visual effects.

5.2. Camera Localization Sequence

We took one sequence (Chess, sequence 2) of the data set of Shotton *et al.* [10], and added a virtual object.

References

[1] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, 2014. 2

[2] G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000. 2

[3] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *TPAMI*, 2003. 2

[4] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *ACCV*, 2012. 2, 3, 4

[5] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, 2011. 2

[6] S. G. Johnson. The nlopt nonlinear-optimization package. 2

[7] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015. 4

[8] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *IJCV*, 2009. 2

[9] J. A. Nelder and R. Mead. A simplex method for function minimization. In *Computer Journal*, 1965. 2

[10] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, 2013. 3, 4

[11] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. H. S. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015. 4

Table 1. Results of our approach per object on the data set of Hinterstoisser *et al.* [4].

Object	2D Proj.	2D B. Box	6D Pose ([4])	6D Pose ([10])	Median X Err.	Median Y Err.	Median Z Err.	Median Rot. Err.
RGB								
Ape	85.2%	98.2%	33.2%	34.4%	0.2cm	0.2cm	2.1cm	6.4°
Bench Vise	67.9%	97.9%	64.8%	40.6%	0.2cm	0.2cm	2.1cm	5.7°
Camera	58.7%	96.9%	38.4%	30.5%	0.3cm	0.4cm	2.7cm	6.7°
Can	70.8%	97.9%	62.9%	48.4%	0.2cm	0.2cm	1.9cm	4.9°
Cat	84.2%	98.0%	42.7%	34.6%	0.2cm	0.2cm	2.1cm	6.4°
Driller	73.9%	98.6%	61.9%	54.5%	0.2cm	0.2cm	2.2cm	4.1°
Duck	73.1%	97.4%	30.2%	22.0%	0.3cm	0.2cm	2.7cm	9.0°
Egg Box	83.1%	98.7%	49.9%	57.1%	0.2cm	0.2cm	1.8cm	4.2°
Glue	74.2%	96.6%	31.2%	23.6%	0.3cm	0.2cm	3.1cm	8.4°
Hole Puncher	78.9%	95.2%	52.8%	47.3%	0.2cm	0.2cm	1.7cm	5.1°
Iron	83.6%	99.2%	80.0%	58.7%	0.2cm	0.2cm	1.4cm	4.2°
Lamp	64.0%	97.1%	67.0%	49.3%	0.3cm	0.2cm	1.8cm	4.7°
Phone	60.6%	96.0%	38.1%	26.8%	0.3cm	0.3cm	3.3cm	6.9°
Average	73.7%	97.5%	50.2%	40.6%	0.2cm	0.2cm	2.2cm	5.9°
RGB-D								
Ape	95.8%	99.7%	98.1%	59.0%	0.1cm	0.1cm	0.2cm	4.4°
Bench Vise	97.3%	99.2%	99.0%	92.9%	0.1cm	0.1cm	0.2cm	2.3°
Camera	98.7%	96.9%	99.7%	92.8%	0.1cm	0.1cm	0.2cm	2.4°
Can	98.6%	99.8%	99.7%	89.6%	0.1cm	0.1cm	0.2cm	2.6°
Cat	97.9%	99.8%	99.1%	80.1%	0.1cm	0.1cm	0.1cm	2.9°
Driller	93.2%	99.4%	100.0%	93.1%	0.1cm	0.1cm	0.3cm	1.8°
Duck	90.5%	100.0%	96.2%	52.1%	0.1cm	0.1cm	0.2cm	4.8°
Egg Box	98.2%	99.1%	99.7%	96.8%	0.1cm	0.2cm	0.2cm	2.2°
Glue	92.8%	100.0%	99.0%	55.1%	0.1cm	0.1cm	0.2cm	4.6°
Hole Puncher	96.1%	99.6%	98.0%	80.3%	0.1cm	0.1cm	0.2cm	3.1°
Iron	97.0%	99.0%	99.9%	96.9%	0.1cm	0.1cm	0.2cm	2.6°
Lamp	92.4%	100.0%	99.5%	91.7%	0.2cm	0.1cm	0.2cm	2.7°
Phone	95.6%	99.8%	99.6%	86.8%	0.2cm	0.1cm	0.3cm	2.6°
Average	95.7%	99.6%	99.0%	82.1%	0.1cm	0.1cm	0.2cm	4.3°

Table 2. Results of our approach per object on the data set of Shotton *et al.* [10].

	5cm 5°				Median Error	
	RGB		RGB-D		RGB	
Scene	Ours	Sparse RGB[10]	Ours	Valentin *et al.* [11]	Ours	PoseNet[7]
Chess	94.9%	70.7%	99.6%	99.4%	1.5cm, 1.3°	32cm, 3.8°
Fire	73.5%	49.9%	94.0%	94.6%	3.0cm, 1.4°	57cm, 7.0°
Heads	48.1%	67.6%	89.3%	95.9%	5.9cm, 3.4°	30cm, 6.1°
Office	53.2%	36.6%	93.4%	97.0%	4.7cm, 1.7°	48cm, 5.1°
Pumpkin	54.5%	21.3%	77.6%	85.1%	4.3cm, 2.1°	49cm, 4.3°
Red Kitchen	42.2%	29.8%	91.1%	89.3%	5.8cm, 2.2°	64cm, 4.2°
Stairs	20.1%	9.2%	71.7%	63.4%	17.4cm, 7.0°	48cm, 7.5°
Average	55.2%	40.7%	88.1%	89.5%	6.1cm, 2.7°	46.9cm, 5.4°

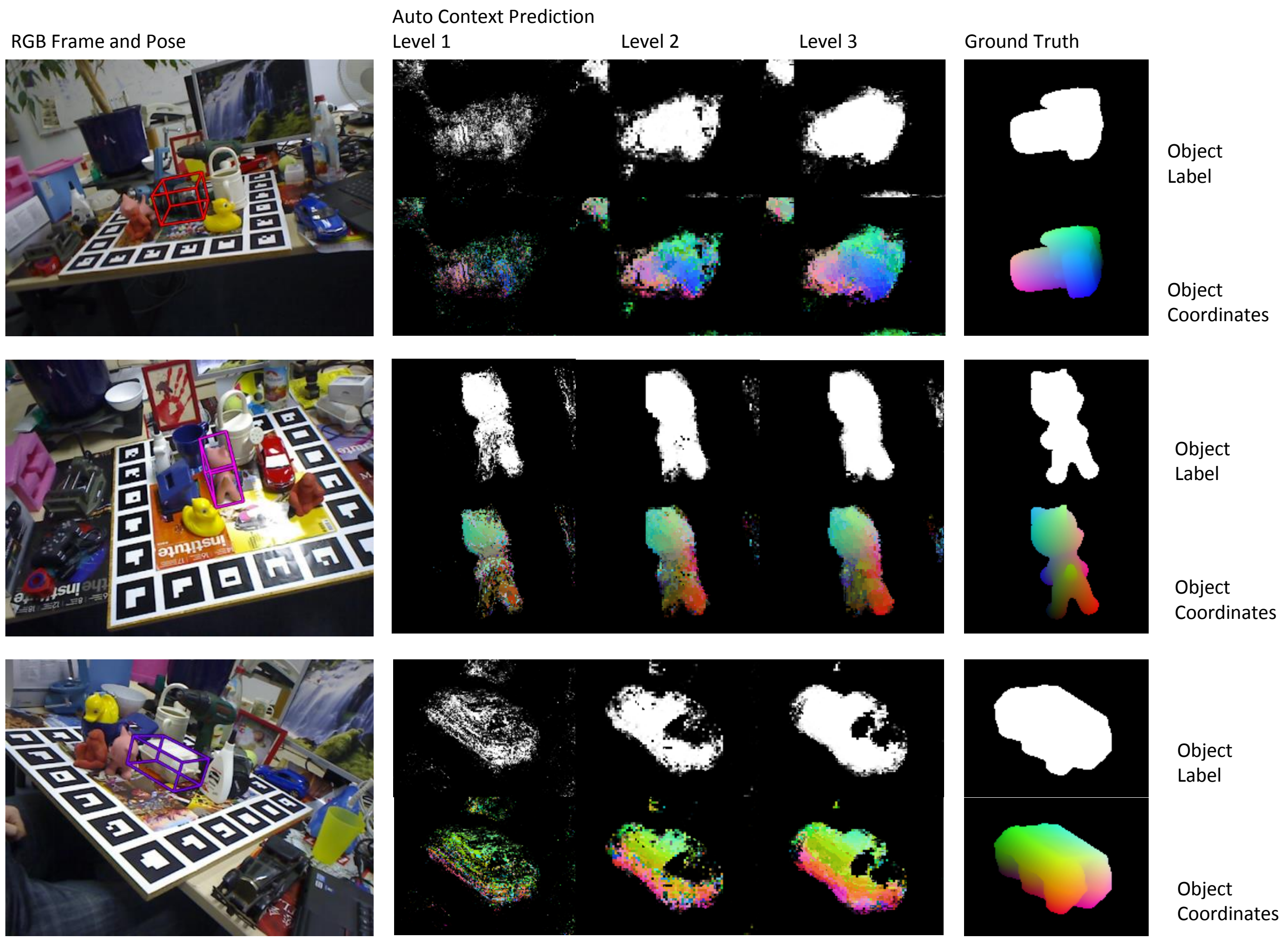

Figure 1. Single object pose estimation results. (Left) An RGB frame with the estimated pose of one object. The bounding box color encodes the object ID. (Center) The prediction of the auto-context random forest at different stack levels zoomed in on the object. The top row shows object probabilities, the bottom row shows object coordinates. Object coordinates are visualized by mapping x/y/z to r/g/b. (Right) Ground truth segmentation and ground truth object coordinates.

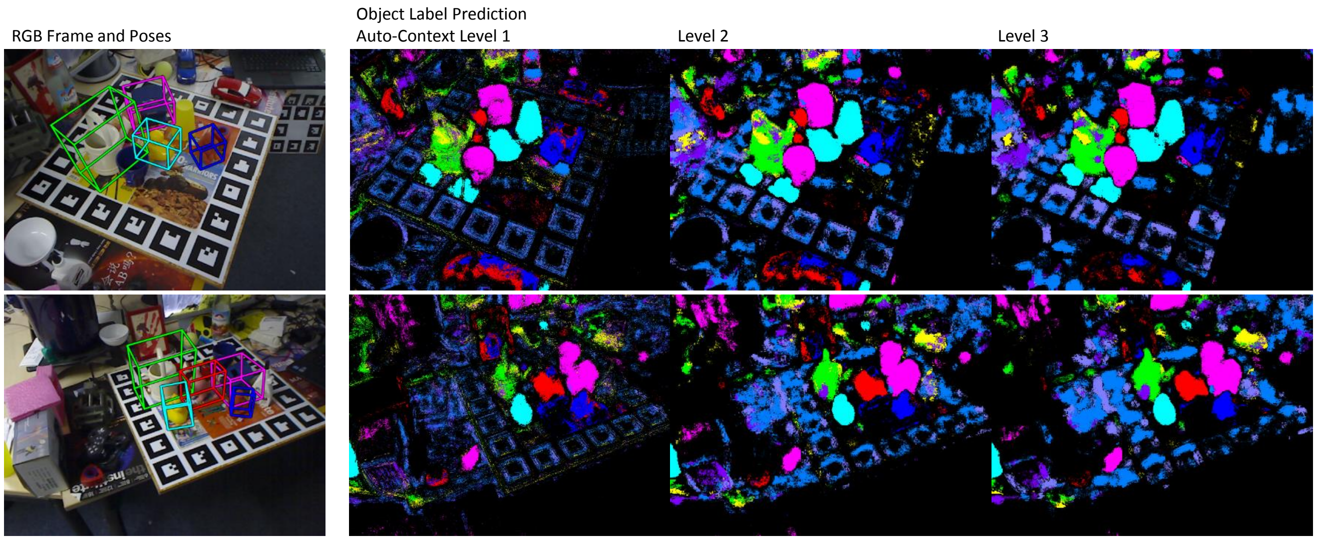

Figure 2. Multi-object pose estimation results. (Left) An RGB frame with the estimated poses of all objects. Only detections with an minimum number of 400 inliers are shown. (Right) The label prediction of the auto-context random forest at different stack levels. Colors encode object IDs.

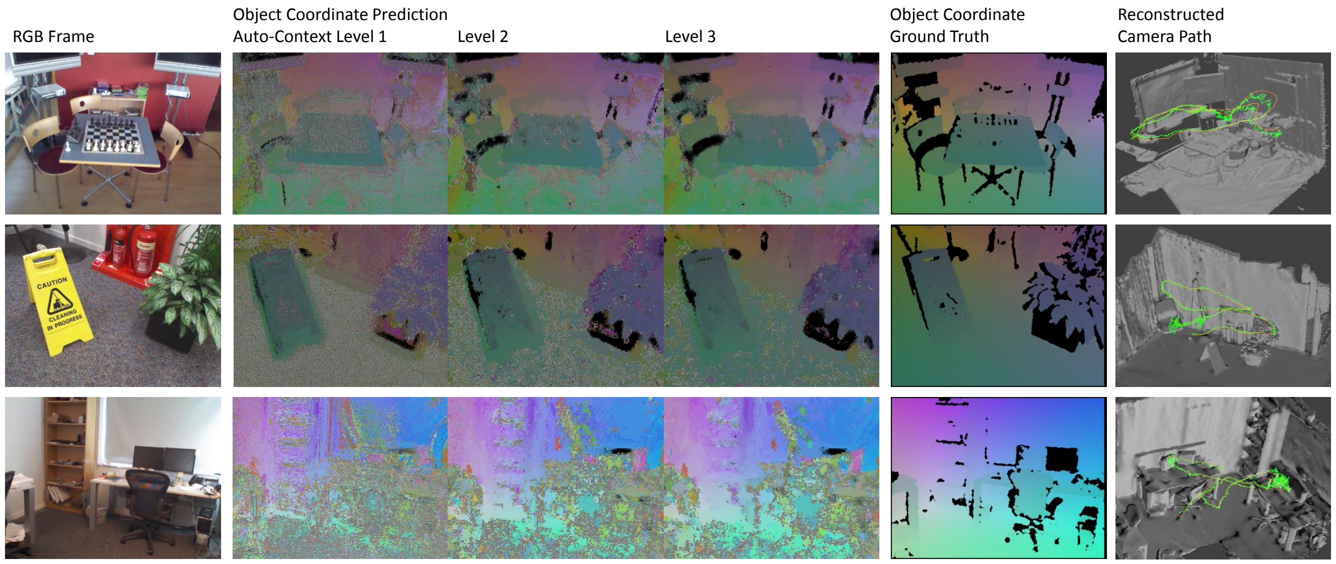

Figure 3. Camera localization results. We show (from left to right) an RGB frame of a room sequence, the object coordinate prediction of different auto-context levels, the object coordinate ground truth, and the estimated camera path (green) for one complete image sequence. We removed some extreme outlier poses, which results in rare gaps in the estimated path. The ground truth camera path (orange) is also shown for comparison.