Supplementary Material: CASAPose: Class-Adaptive and Semantic-Aware Multi-Object Pose Estimation

Niklas Gard[1]
niklas.gard@hhi.fraunhofer.de
Anna Hilsmann[1]
anna.hilsmann@hhi.fraunhofer.de
Peter Eisert[1,2]
peter.eisert@hhi.fraunhofer.de

[1] Fraunhofer HHI
Einsteinufer 37,
Berlin, Germany

[2] Institute for Computer Science
Humboldt University of Berlin,
Berlin, Germany

We provide visual results of CASAPose in Section A. Section B contains further comparisons with the state of the art as well as two more experiments. Finally, Section C lists additional implementation details.

A Visual Results

Fig. 1 shows estimated poses for three example images from Occluded LINEMOD (LM-O) [1] using CASAPose trained for 8 objects. Similarly, Fig. 2 gives an impression of example results using a model trained for 13 objects on our *pbr* test scene [3]. Example results of CASAPose for HomebrewedDB (HB) [4] are shown in Fig. 3.

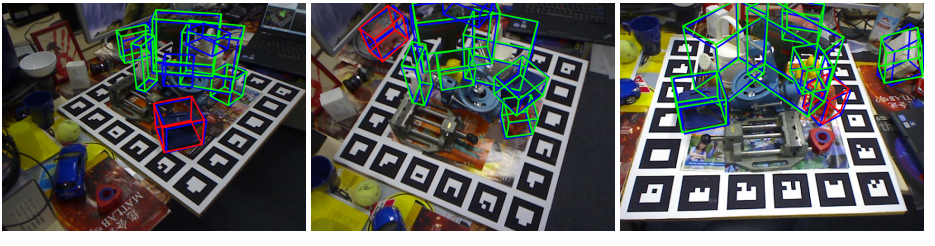


Figure 1: Example results of CASAPose for LM-O, with bounding boxes for correctly estimated poses in green, incorrect poses in red, and ground truth poses in blue (ADD/S metric).

A.1 Effect of Guided Operations

The ablation study discovered that semantic guidance improves the estimated vector fields and the accuracy of pose estimation. The effect is best seen in direct visual comparison.

© 2022. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

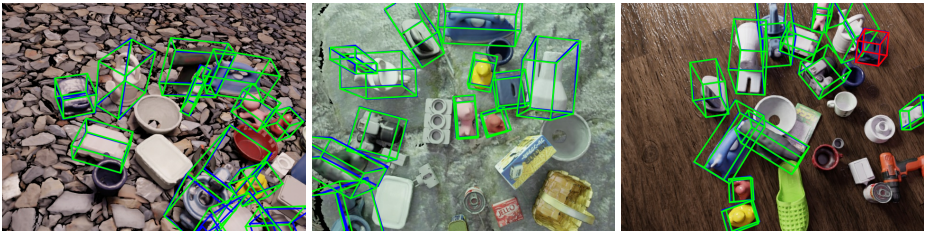


Figure 2: Example results of CASAPose for *pbr* images excluded during training, with bounding boxes for correctly estimated poses in green, incorrect poses in red, and ground truth poses in blue (ADD/S metric).

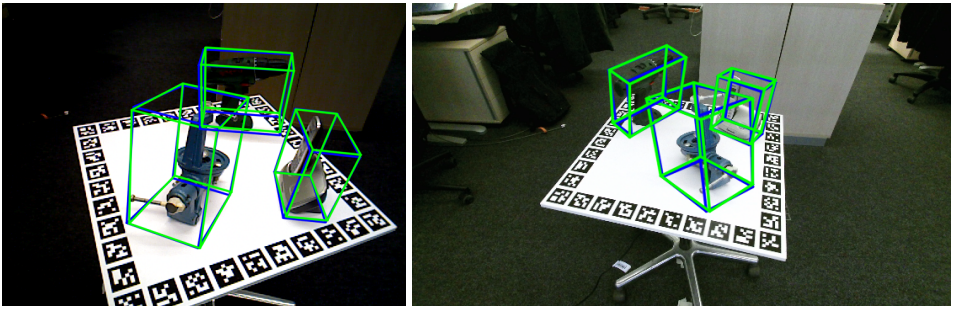
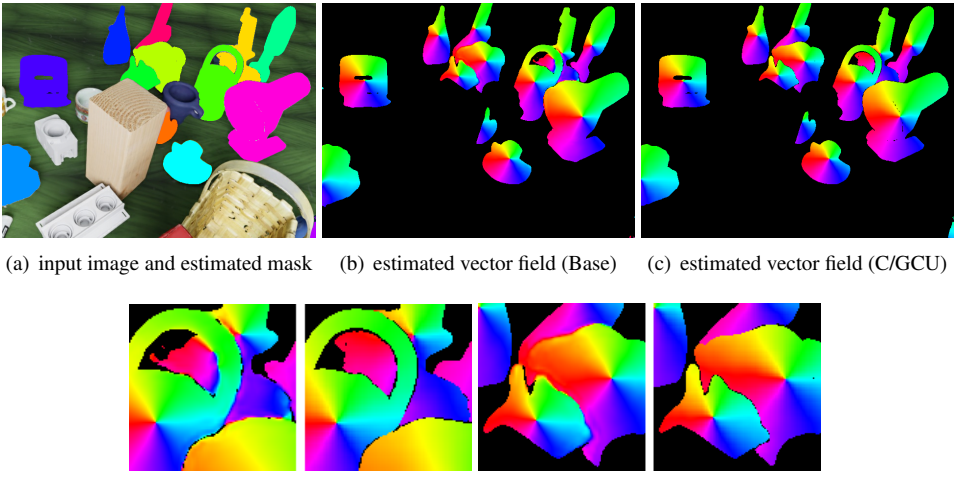


Figure 3: Example results of CASAPose for HB, with bounding boxes for correctly estimated poses in green and ground truth poses in blue (ADD/S metric). The left image is captured with the PrimeSense Carmine camera; the right image is captured with Microsoft Kinect 2.

Fig. 4 shows the enhancement exemplified for an image from the *pbr* dataset using colour coded vector fields for visualisation. It shows the vector fields for the first keypoint, which is always located in the centre of each object. Comparing the output of a model without the guided decoder in Fig. 4(b) with the output of a model with object-aware convolutions and object-aware upsampling in Fig. 4(c), there is a clear improvement in vector fields, especially in regions where objects overlap. In fact, we have observed that a network without semantic guidance is not even able to produce perfectly separated vector fields when it heavily overfits only a few images. Using CLADE alone without semantic guidance already improves the quality of the vector fields per object due to the object-specific parameters (see Table 4 of the main paper), but a clear separation as in Fig. 4(c) can only be achieved in combination.

A.2 Characteristics of the Learned Confidence Maps

Fig. 5 shows the estimated vector fields and confidence maps for an image from LM-O using the 8-object model. The estimated 2D locations are highlighted by a white circle. The confidence values are normalised inside each semantic mask for clearer presentation. For the first keypoint (Fig. 5(b)), which is always in the centre of the object, the confidence is relatively constant in each mask, indicating that it is easy for the network to predict this point with high accuracy. In Fig. 5(c) and 5(d), it can be seen that the regions where the network predicts high confidence are often spatially close to the actual keypoint location. For example,



(a) input image and estimated mask (b) estimated vector field (Base) (c) estimated vector field (C/GCU)

(d) detail comparison 1 (e) detail comparison 2

Figure 4: Visual comparison of the estimated vector fields for a network with (*C/GCU*) and without (*Base*) the semantically guided operations. In the detail comparisons, *Base* is on the left, while *C/GCU* is on the right.

for the tip of the tail of 'cat' in Fig. 5(c), it is logical that the best prediction of the location can be made nearby. Moreover, for example, 'ape' in Fig. 5(d) shows that the model predicts high reliability and thus computes the 2D position of a keypoint mainly from pixels near the object silhouette. Especially for non-textured objects, the silhouette provides important information about the orientation of the object. It seems appropriate that the vectors near the contour can be estimated with higher precision.

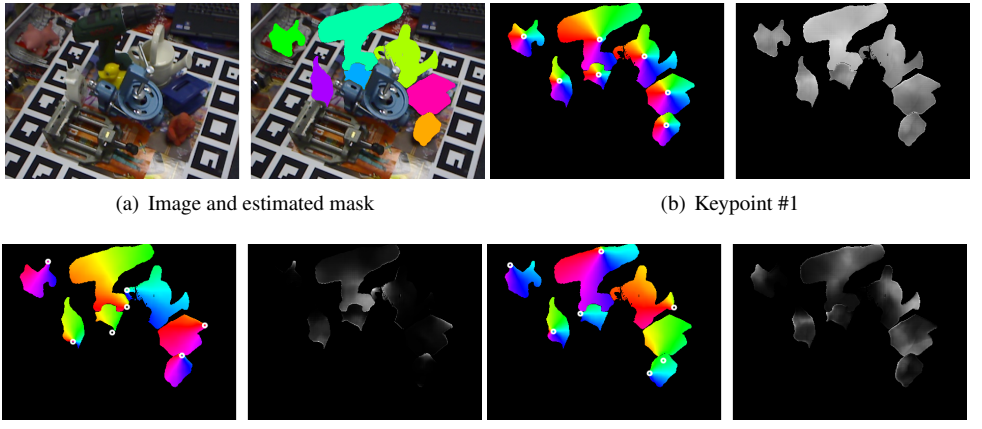
B Additional Experiments

B.1 BOP Challenge Evaluation

In the BOP Challenge 2020 [3] multiple approaches submitted results for several pose estimation datasets, including LM-O with synthetic training. The results presented are in most cases significantly improved by subsequently inserted changes compared with the results from the original publications. We also evaluated our results against the BOP benchmark. It calculates an accuracy called Average Recall (**AR**), which is the average of the results for three pose error functions, Maximum Symmetry-Aware Projection Distance (MSPD), Maximum Symmetry-Aware Surface Distance (MSSD), and Visible Surface Discrepancy (VSD). Further details can be found in [3].

Table 1 lists the results of our procedure with this metric. CASAPose$_8$ is our final result for the 8-object case from the main paper. EPOS [2, 3], the only other single-stage multi-object method achieved an **AR** of 54.7, slightly higher than CASAPose. CASAPose$_{8*}$ trained with minimally different hyper parameters (increasing λ_4 from 0.007 to 0.01), again achieves a slightly superior result, showing that both methods are similarly accurate. Still, our method is multiple times faster (468ms vs. 37 ms) [1]. CDPNv2 [3, 7], the best method using only

[1]The difference is so significant, that it can also not be explained by our faster evaluation GPU.



(a) Image and estimated mask (b) Keypoint #1

(c) Keypoint #4 (d) Keypoint #5

Figure 5: Estimated vector fields and confidence maps for three out of nine keypoints.

rgb images and no additional refinement (EPOS would be the second best method on LM-O with these properties), reaches an **AR** of 62.4, but trains a single network for every object. They make numerous extensions to their original approach (adding more complicated domain randomisation and a more powerful backbone) that would potentially improve our method as well, but are out of scope of this paper. The innovations of our paper to convert a multi-stage (one network per object and bounding box detector) approach into a single-stage (one network for all objects without need for a bounding box detector) approach could be applied analogously to their method.

Arch.	DNN	AR	AR_{MSPD}	AR_{MSSD}	AR_{VSD}
EPOS [7, 8]	1/set	54.7	75.0	50.1	38.9
CASAPose_8	1/set	54.2	74.3	49.4	39.0
CASAPose_{8*}	1/set	55.4	75.3	50.8	40.2
CASAPose_{2x4}	2/set	57.4	77.1	52.9	42.1

Table 1: Comparison of different variants of our method using the BOP benchmark on LM-O with EPOS [7] in BOP configuration [8, 11]. CASAPose_{2x4} is the *'4 Obj. 2x'* result from Table 5 of the main paper using 2 notworks, each for 4 objects.

B.2 Ablation Study: Guided Decoder

We tested different versions of the semantically guided decoder for the 13-object configuration trained with *DKR* (Table 2). The first variant *C/GU* uses only guided upsampling and no guided convolution. Compared with CLADE (*C*) alone, this does not bring an improvement, since the increased accuracy during upsampling is cancelled out by the following regular convolution, which does not take the masks into account. Adding guided convolutions in the first 3 or 4 of 5 decoder blocks (*C/GCU3*, *C/GCU4*) improves the average **2DP** and the average **ADD/S**. Between *C/GCU3* and *C/GCU4* no clear difference is visible. Comparing

	LM-O		LM	
	2DP	ADD/S	2DP	ADD/S
C	51.4	28.9	93.6	64.7
C/GU	50.7	30.0	93.9	62.9
C/GCU3	52.0	32.1	93.7	65.5
C/GCU4	**52.7**	31.9	93.5	64.9
C/GCU5	51.5	**32.7**	**93.8**	**68.1**

Table 2: Comparison of different versions of the semantically guided decoder using the 13-object model with *DKR*.

	2DP	ADD/S
$PV_{RANSAC*}$	49.2	26.7
PV_{RANSAC}	50.4	30.8
LS_{All}	45.3	29.7
$LS_{2ndComp.}$	7e-3	2e-3
$LS_{1stComp.}$	**51.5**	**32.7**

Table 3: Comparison of different variants of 2D keypoint calculation using the 13-object model (*C/GCU5*) on LM-O.

the final model *C/GCU5* (with guided convolutions in all decoder blocks) with *C/GCU4*, the **2DP** decreases by 0.6% on average, while an increase of 4.1% of **ADD/S** outweighs this. This makes this architecture the best among the tested ones.

B.3 Influence of Keypoint Regression

Table 3 compares different variants of the calculation of 2D keypoint positions. $LS_{1stComp.}$ is the variant used in our final model. It applies *DKR* on the largest connected component of each object class and clearly outperforms RANSAC voting (PV_{RANSAC}) [8] used with the same trained model. Interestingly, if a network learns to estimate confidence maps with *DKR* during training, also the results of the RANSAC voting improve (PV_{RANSAC} compared with $PV_{RANSAC*}$). This suggests that least squares optimisation over all vectors in a region during training also improves the global accuracy of the estimated vectors. Applying *DKR* on a complete mask without connected component filtering (LS_{All}) deteriorates the performance, indicating that potential clutter in the estimated semantic masks should be removed before calculating the 2D positions. We tested *DKR* on the second largest connected component $LS_{2ndComp.}$ and see that it nearly never leads to a correct pose. So, at least for the case where only one object per class is visible, using only the largest connected components is very suitable. A proposal for adaptation to multi-instance scenarios is given in the main paper.

C Additional Details

C.1 Differentiable Keypoint Regression

The Differentiable Keypoint Regression uses a weighted Least Squares intersection of lines calculation, incorporating confidence scores as weights, as it is described e.g. in [4]. One system is constructed per keypoint per object. All systems are solved in parallel using `tf.linalg.pinv` to calculate the Moore-Penrose pseudo-inverse. *DKR* uses the *softplus* function to translate from the network output to the weights of the Least Squares calculations. It is a smooth approximation of the *ReLU* function that constraints the output of the network to be non negative. Compared with *sigmoid*, it allows to predict weights greater than 1. During training, we add a regularisation term to avoid drift of the Least Squares weights for *DKR* towards zero or infinity. The mean value in the foreground regions of each output map is $\ell 1$ regularised to be close to a constant value of 0.7.

C.2 Hyperparameter Choices

The losses $\mathcal{L}_{Seg}$, $\mathcal{L}_{Vec}$, $\mathcal{L}_{PV}$, and $\mathcal{L}_{Key}$ are weighted with the factors λ_{1-4}. Previous work weighted $\mathcal{L}_{Seg}$ and $\mathcal{L}_{Vec}$ equally ($\lambda_1 = \lambda_2 = 1.0$) [8], or additionally added $\mathcal{L}_{PV}$ as a regularizer with much smaller weight [10]. In tests without *DKR* and $\mathcal{L}_{Key}$, we determined 0.015 as a suitable choice for λ_3. This is larger than the recommendation of [10], but leads to stable convergence in our case. A reduction of the influence of $\mathcal{L}_{Vec}$ ($\lambda_2 = 0.5$) after including $\mathcal{L}_{Key}$ preserves the balance between segmentation (λ_1) and vector field prediction (λ_{2-4}). In summary, the weights were $\lambda_1 = 1.0$, $\lambda_2 = 0.5$, $\lambda_3 = 0.015$ and $\lambda_4 = 0.007$.

As reported in Section B.1, we also trained a full model using $\lambda_4 = 0.01$ and observed a slight accuracy increase for LM-O (8 objects). However the 13-object model (LM) did not converge as good in this setting.

For models with two decoders (*C* and *C/GCU*), the calculation of $\mathcal{L}_{Vec}$ and $\mathcal{L}_{PV}$ evaluates only those locations where the estimated segmentation matches the true segmentation, which in our experience slightly improves the training result.

C.3 Further Details

- The Farthest Point Sampling (FPS) algorithm is used to calculate the 3D locations of the keypoints [8]. The keypoint set is initialised by adding the object centre. The 3D models of HB originate from a different 3D scan and have their origin in a different location than the models of LM. We aligned them with the Iterative Closest Point (ICP) algorithm and calculate a fixed compensation transformation for each model. It is applied to the 3D keypoints to make a comparison with HB's ground truth.

- Pose estimation uses OpenCV's `cv::solvePnPRansac` with EPnP [6] followed by a call of `cv::solvePnP` with `SOLVEPNP_ITERATIVE` and the previous pose as `ExtrinsicGuess`.

- During training, we use scenes 0-48 from the synthetic *pbr* LINEMOD images from [3] resulting in 49000 training images. Scene 49 (1000 images) is kept for testing and is used in the ablation study.

- The experiments were conducted using Tensorflow 2.9 and use ADAM optimiser [5]. Our custom layers use Tensorflow's `tf.function(jit_compile=True)` for acceleration.

References

[1] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *Proc. ECCV*, 2014.

[2] Tomas Hodan, Daniel Barath, and Jiri Matas. Epos: Estimating 6d pose of objects with symmetries. In *Proc. CVPR*, 2020.

[3] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. Bop challenge 2020 on 6d object localization. In *Proc. ECCV Workshops*, 2020.

[4] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In *Proc. ICCV Workshops*, 2019.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICRL*, 2015.

[6] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[7] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proc. ICCV*, 2019.

[8] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proc. CVPR*, 2019.

[9] Johannes Traa. Least-squares intersection of lines. *University of Illinois Urbana-Champaign (UIUC)*, 2013.

[10] Xin Yu, Zheyu Zhuang, Piotr Koniusz, and Hongdong Li. 6dof object pose estimation via differentiable proxy voting regularizer. In *Proc. BMVC*, 2020.

[11] BOP: Benchmark for 6D Object Pose Estimation. Submission: EPOS-BOP20-PBR/LM-O. https://bop.felk.cvut.cz/sub_info/1382/, 2020. [Online; accessed 27-July-2022].