Towards Meaningful Uncertainty Information for CNN based 6D Pose Estimates

Jesse Richter-Klug[0000−0002−6860−5992] and Udo Frese[0000−0001−8325−6324]

University Bremen, 28359 Bremen, Germany
{jesse,ufrese}@uni-bremen.de

Abstract. Image based object recognition and pose estimation is nowadays a heavily focused research field important for robotic object manipulation. Despite the impressive recent success of CNNs to our knowledge none includes a self-estimation of its predicted pose's uncertainty.
In this paper we introduce a novel fusion-based CNN output architecture for 6d object pose estimation obtaining competitive performance on the YCB-Video dataset while also providing a meaningful uncertainty information per 6d pose estimate. It is motivated by the recent success in semantic segmentation, which means that CNNs can learn to know what they see in a pixel. Therefore our CNN produces a per-pixel output of a point in object coordinates with image space uncertainty, which is then fused by (generalized) PnP resulting in a 6d pose with 6×6 covariance matrix. We show that a CNN can compute image space uncertainty while the way from there to pose uncertainty is well solved analytically. In addition, the architecture allows to fuse additional sensor and context information (*e.g.* binocular or depth data) and makes the CNN independent of the camera parameters by which a training sample was taken.

Keywords: 3D object pose estimation · uncertainty · gPnP.

1 Introduction

1.1 Motivation

Convolutional Neural Networks (CNNs) have immensely improved the capability of state of the art computer vision. This development started with qualitative questions, such as "What is in the image?" [10], over "Where is something in the image?" [17] to metrical questions, such as "Where is an object in space?" [22,23]. The latter is of particular interest for robotics, because a robot normally needs a 6d pose (position and orientation) to grasp an object.

The success of Deep Learning is accompanied by an uneasiness on the lack of transparency, which has many aspects. One aspect is, that it is hard to understand how CNNs come to the solution they output. Another aspect is, that one would desire a measure of uncertainty that indicates how precise and/or reliable the output is. Towards this goal, this paper provides such a measure as a 6×6 covariance matrix for 6d object poses estimated by a CNN.

Such uncertainty information offers many opportunities. It could be used to

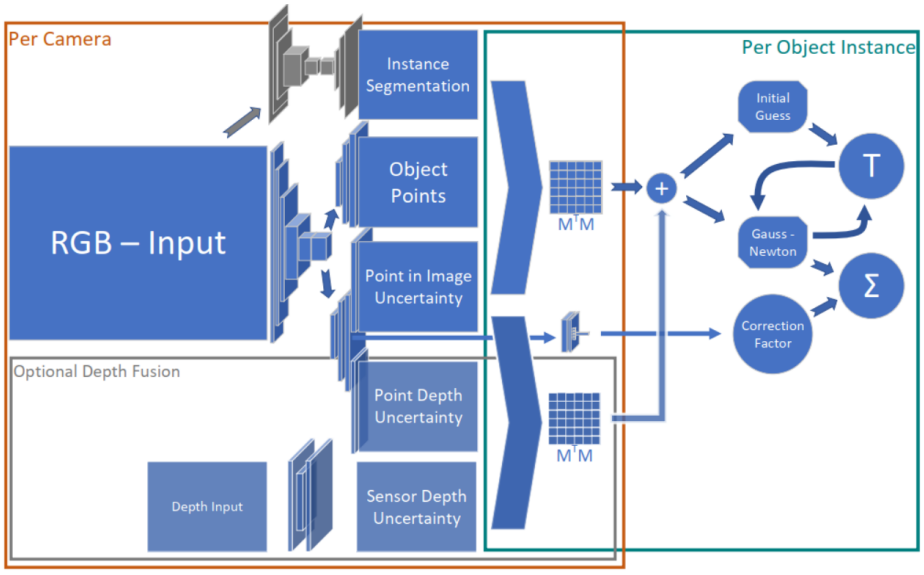


Fig. 1. Proposed architecture overview (cf. Section 3): CNNs take the image and optionally depth data and provide per-pixel instance segmentation, object points and uncertainties, as well as a per-object overall uncertainty correction factor (Section 4). This data is aggregated into a 13×13 matrix $M^T M$ and fused by a gPnP algorithm (Section 5) to a pose estimate T with covariance Σ.

- safeguard action, *e.g.* only grasp if the object pose is precise enough,
- trigger action *e.g.* look from a different perspective, if necessary,
- fuse several object observations with established probabilistic methods [21],
- fuse with prior information, *e.g.* that objects often rest on a horizontal plane.

1.2 Contribution

Towards this goal, we propose here an architecture (Fig. 1) where an encoder/decoder CNN with bypass connections predicts for every pixel belonging to an object "what is seen there" with uncertainty. Concretely it predicts the 3d object point belonging to that pixel in object coordinates as well as the 2D uncertainty of the predicted object point in the image space.

These 3D-2D pairs with uncertainty are then input to a Perspective-n-Point (PnP) solver that computes the pose estimate with uncertainty using probabilistic methods and the camera calibration. Actually, the solver handles generalized PnP (gPnP) [9] problems, so available depth information and binocular cameras can be included.

This approach follows the idea that analytical methods are better at multi-view geometry because it is well understood, whereas machine learning is better at recognition, because it is not well understood what defines an object in a real

Fig. 2. Object orientation is not even approximately a translation invariant function of the image: In the middle, there is a tube in front of the camera. The right tube is translated, *i.e.* has the same orientation but looks very different. The left tube is rotated around the camera, *i.e.* has a different orientation but looks almost identical.

image. In particular, by construction gPnP handles the effects of camera calibration and occlusion on the estimate and its uncertainty structurally correctly.

2 Related Work

2.1 CNN based Pose Estimation

From the perspective of this paper, the CNN based pose estimation methods can be categorized according to how the pose is coded in the CNNs output.

Some practical approaches, *e.g.* [25] in the Amazon Picking Challenge use CNNs to segment an object in RGB data and then perform ICP on the segmented depth data, however this ignores most of the information in the RGB data.

Early pose estimation CNNs, such as [18] for outdoor camera pose estimation directly predicted a vector and quaternion. So some object pose CNNs, such as PoseCNN [23], SSD6D [5], AAE [19] and Periyasamy *et al.* [13] evolved from object detection CNNs by adding a pose output stage. Usually they predict the 2D object center (relative to the pixel, often with voting), distance (directly or from bounding box size) and orientation (discretized or quaternion). This wrongly treats orientation as a translation invariant function of the image which is not true as we show in Fig. 2.

Newer works assume, that CNNs can predict image-space quantities best. Examples are YOLO6D [20], BB8 [15], Crivallaro *et al.* [2], Oberweger *et al.* [12] and DOPE [22]. These predict the image position of *e.g.* bounding box corners which are then fed into a PnP-solver (*e.g.* EPnP [11]). This also isolates the CNN from camera calibration handled by the PnP stage.

One can take this idea further and let every pixel predict, which object point it sees. Brachmann *et al.* [1] did this using random-forests, very recently DPOD [24] with CNNs and we also follow this line. The approach handles occlusions well [24, Table 2], arguably because as long as visible object points are correct, PnP tolerates missing points. Crivallaro *et al.* [2] predict points for object parts and Oberweger *et al.* [12] limit the receptive field for the same reason.

AAE [19] considers the problem of symmetries, which come in continuous form (*e.g.* a bottle), discrete form (*e.g.* a cube) and as view dependent ambiguity (*e.g.* a cup with occluded handle). They solve it with a denoising autoencoder that discovers the appropriate rotation representation for the object.

Among the mentioned approaches only [2] considers uncertainty. By propagating an assumed image noise through the Jacobian of the CNN, they predict 2D covariances for the control points. They see these only as a tool to improve the fusion result and neither validate them empirically nor output a pose covariance.

2.2 PnP and Variants

PnP, *i.e.* computing a pose from n 2D-3D point correspondences, is a classical geometric vision problem. Lepetit *et al.* [11] condensed the n points into a fixed size 12×12 matrix, by expressing them as a convex combination of four so-called control points. Their EPnP algorithm globally finds the best fitting control points with four cases depending on the number of small eigenvalues.

Kneip [9] extends EPnP to rays not intersecting in a single point, *e.g.* from multiple cameras (generalized or gPnP). The input is condensed into a 12×12 matrix and a 12 vector and solved using Gröbner bases considering 6 cases. This and other geometric vision algorithms are published in the opengv library [8].

Zheng [26] casts PnP as quaternion minimization, solved by a Gröbner basis.

These algorithms provide global solutions making them fairly complex. Unlike the iterative Gauss-Newton, they don't provide uncertainty information.

The PnP solution in [2] fuses with a pose prior of 9 Gaussians, effectively by Gauss-Newton on 9 initial guesses, while [12, 15, 20, 22, 24] use EPnP black-box.

2.3 Gaussian distributions and Gauss-Newton on $SE(3)$

The space of poses $SE(3) \subset \mathbb{R}^{4\times4}$ is a manifold and special care is needed to define Gaussians and perform least-squares there. We use $\boxplus$-manifolds [3] that encapsulate the manifold structure into an operator $\boxplus : SE(3) \times \mathbb{R}^6 \to SE(3)$. It takes a pose and changes it according to a vector in the tangential space, usually using the exponential map. A converse operator $\boxminus : SE(3) \times SE(3) \to \mathbb{R}^6$ axiomatized by $T_1 \boxplus (T_2 \boxminus T_1) = T_2$ gives the vector that goes from one pose T_1 to another T_2. These operators allow also to define Gaussians in $SE(3)$.

$$T \boxplus \delta = T \exp\begin{pmatrix} 0 & -\delta_3 & \delta_2 & \delta_4 \\ \delta_3 & 0 & -\delta_1 & \delta_5 \\ -\delta_2 & \delta_1 & 0 & \delta_6 \\ 0 & 0 & 0 & 0 \end{pmatrix} \overset{\text{linearized}}{\approx} T\begin{pmatrix} 1 & -\delta_3 & +\delta_2 & \delta_4 \\ +\delta_3 & 1 & -\delta_1 & \delta_5 \\ -\delta_2 & +\delta_1 & 1 & \delta_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

$$T_2 \boxminus T_1 = \left(L_{32}, L_{13}, L_{21}, L_{14}, L_{24}, L_{34} \right)^T, \quad L = \log(T_1^{-1} T_2) \overset{\text{linearized}}{\approx} T_1^{-1} T_2 - I \tag{2}$$

$$\mathcal{N}(\mu, \Sigma) \quad = \mu \boxplus \mathcal{N}(0, \Sigma), \quad \mu \in SE(3), \Sigma \in \mathbb{R}^{6\times6} \tag{3}$$

$$\mathcal{N}(\mu, \Sigma)(x) \approx |2\pi\Sigma|^{-1/2} \exp\left(-\tfrac{1}{2}(x \boxminus \mu)^T \Sigma^{-1} (x \boxminus \mu)\right) \tag{4}$$

In this framework a squared residual $\|f(T)\|^2$ on $T \in SE(3)$ is minimized with Gauss-Newton [14] by parameterizing the increment with $\boxplus$ in each iteration:

$$\underset{T\in SE(3)}{\arg\min} \ \|f(T)\|^2 = \lim_k T_k, \quad T_{k+1} = T_k \boxplus \overset{\text{linearized}}{\underset{\delta\in\mathbb{R}^6}{\arg\min}} \ \|f(T_k \boxplus \delta)\|^2 \tag{5}$$

3 Approach

Figure 1 shows our approach: A CNN recognizes as first stage what is in the images, segments the objects and predicts for every pixel u_i belonging to an object which object point p_i^O this is. It also predicts the uncertainty of this

information, expressed as a covariance $C_i \in \mathbb{R}^{2\times 2}$ in image space. These pieces of information are then fused by the second stage (gPnP) to obtain an object pose with covariance derived from the point covariances.

Actually the uncertainty orginates from p_i^O while the pixel coordinates u_i are exact, however the CNN operates on the image, so presumably it can more easily express image uncertainty (*e.g.* along an image edge). Furthermore, only this way fits into the mathematical structure of gPnP.

To handle multiple cameras, the unknown object pose $T := T_O^R$ is expressed in a reference frame (R). A known transformation $T_R^{C_i}$ locates the reference frame in the frame of the camera to which pixel u_i belongs. We omit the index at C_i in the following. Let p_i^C denote p_i^O in camera-coordinates, u_0 the image center, and f the focal length. With this notation, the camera equation with noise η_i defines the interface between the CNN and gPnP stages:

$$p_i^C = T_R^C T_O^R p_i^O, \quad T := T_O^R, \quad u_i = u_0 + f\frac{p_{i12}^C}{p_{i3}^C} + \eta_i, \quad \eta_i \sim \mathcal{N}(0, C_i) \tag{6}$$

$$C_i = (W_i^T W_i)^{-1} \quad \Longrightarrow \quad W_i \left(u_i - u_0 - f\frac{p_{i12}^C}{p_{i3}^C}\right) \sim \mathcal{N}(0, I_2) \tag{7}$$

The CNN provides C_i indirectly as a 2×2 weight matrix W_i. This should help, as the rows of W_i describe independent image-directions of information with a larger norm meaning more information. In particular, (7) ensures positive semi-definiteness and facilitates expressing anisotropic situations, *e.g.* at image edges.

Contrary to the usual assumption in a PnP stage (cf. 5), different p_i^O are not stochastically independent. Therefore the CNN stage predicts a correction factor w_{oj} per object instance j that upscales the computed covariance Σ_{gPnP} assuming every pixel carries the same percentage w_{oj}^{-1} of "new" information.

$$\Sigma = w_{oj}\Sigma_{\mathrm{gPnP}} \tag{8}$$

The proposed method can also incorporate per-pixel depth information z_i if available into the gPnP fusion with a depth-pendant to (6):

$$z_i = p_{i3}^C + \eta_{di}, \quad \eta_{di} \sim \mathcal{N}(0, w_{di}^{-2}), \quad w_{di} = (w_{doi}^{-2} + w_{dsi}^{-2})^{-1/2} \tag{9}$$

$$\Longrightarrow \quad w_{di}\left(z_i - p_{i3}^C\right) \sim \mathcal{N}(0, 1) \tag{10}$$

The noise η_{di} has two sources: The main RGB-CNN gives a weight w_{dpi} describing the noise in p_i^O. A small depth-CNN gives a weight w_{dsi} describing the noise in z_i, identifying *e.g.* invalid depth data. Both are combined by adding variances. This way the depth data does not contribute to the recognition but is integral part of the gPnP-fusion stage not a separate postprocessing as in PoseCNN.

In the gPnP stage the measurement equations (6) and (9) are converted into 13×13 matrices and added, thereby condensing all information into a globally valid fixed size matrix $M^T M$ without linearization. We obtain an initial guess by removing the translation with Schur-complement and optimizing the rotation-only result on a finite set of 22.5°-spaced rotations. This guess is refined with Gauss-Newton on the $SE(3)$ manifold of poses, giving the pose T and the pose covariance Σ_{gPnP} as the inverse of Gauss-Newton's pseudo-Hessian. $\mathcal{N}(T, \Sigma)$, in the sense of (3), is the then final posterior distribution.

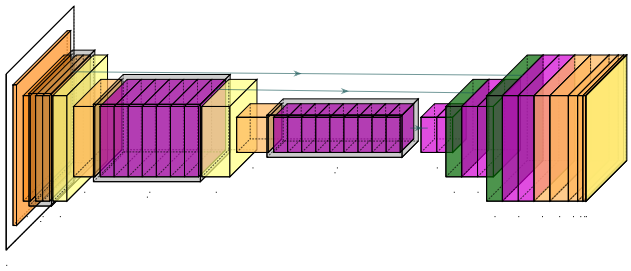


Fig. 3. CNN layers, where outputs are ordered according to Fig. 1: Shades from yellow to orange denote 1×1, 3×3 or 5×5 convolutions; magenta, pairs of 1×1 followed by 3×3 convolutions; green, upsampling; blue, global pool and fully connected layers. All layers with trainable weights are followed by SELU [7] units. Output layers have no activation. For further details see `https://github.com/JesseRK/6D-UI-CNN-Output`.

4 CNN Details

The main CNN consists of an encoder-decoder DenseNet-like [4] structure with horizontal connections. Instead of batch-norm-ReLU we exclusively use SELU [7] activations. There are a shared encoder path and two different but identical decoder paths for object points (p_i^O) and uncertainty predictions. The latter forks into heads for W_i, w_{doi} and w_{oj}. All output images have a quarter (x and y) of their input resolution. More details are shown in Fig. 3, however we see the overall architecture, not the layer details, as the paper's contribution.

Instance segmentation can be performed with an arbitrary method since it is independent of the core object estimation algorithm, and was not the main focus here. For prediction, we use the segmentation masks as they are calculated by PoseCNN [23] and omit further description regarding the segmentation procedure. During training we use ground truth segmentation to prevent distracting dependencies. In general we try to prevent those by training the CNN in three stages as declared in Section 4.1. In Section 4.2 we describe the secondary CNN which is only used in case of depth fusion to estimate the sensor's uncertainty.

4.1 Loss Function and Learning Procedure

Every pose estimation unit (i.e. all except those for segmentation) receives a gradient only if it points at an object that should be learned (i.e. the loss function gets masked with the ground truth segmentation). This allows the network to e.g. predict a p_i^O under the condition that u_i maps to an object point but leaves the decision if this actually is the case to the segmentation. Therefore all outputs are masked by the segmentation before they are used as well.

Currently, we train one network per object as DOPE [22] does. We use `Adam` [6] with the `amsgrad` expansion [16] as optimizer and augment our input images with contrast, Gaussian and brightness noise.

Object Points The p_i^O are exclusively learned first with a mean squared error loss against ground truth $p_{i\,\mathrm{true}}^O$, which can be obtained by rendering the corresponding model with the ground truth pose. It is supplied with the YCb-data set and for rendered images we obtained it from the depth image and ground truth pose. Each axis is separately scaled to the range $[-1, 1]$ for the CNN and the original scale restored before gPnP.

Pixelwise Uncertainties The uncertainty weights W_i and w_{doi} are trained with a negative log likelihood (NLL) loss from the Gaussians they represent.

$$\sum_i -\log \mathcal{N}(0, (W_i^T W_i)^{-1})(\eta_i) - \log \mathcal{N}(0, w_{doi}^{-2})(\eta_{doi}), \tag{11}$$

The residuals η_i and η_{doi} are obtained from (6) and (8) resp. by setting $T = T_{\mathrm{true}}$. The NLL becomes minimal when the residual distribution corresponds to the weight-defined covariance.

After freezing the encoder as well as the respective p_i^O decoder we trained W_i and w_{doi} together as a second training stage.

Object Uncertainty The per object correction factor w_{oj} for the uncertainty obtained by gPnP is also trained with an NLL loss

$$-\log \mathcal{N}(T, \Sigma)(T_{\mathrm{true}}), \tag{12}$$

here of the ground truth pose in the predicted Gaussian posterior. (4) is used, since it is a $\boxplus$-manifold Gaussian.

As third stage, all previously trained layers were frozen and only the w_{oj}-head trained. This training involves the gPnP part. To stabilize the training, Gauss-Newton uses the ground truth pose as initial guess. This approach is sound for training, because the initial guess does not influence the limit, as long as it stays in the basin of convergence for that limit value.

4.2 Depth Sensor Uncertainty CNN

To predict the depth sensor uncertainty we use a separate small network (cf. Fig. 1, sec. 3), which consists of seven SELU-activated convolutions in a ResNet-fashion (up to 32 channels) with one down and one up sampling layer. The only input for this network is the depth image. Since the resulting weights are later combined with the w_{doi} the input and output are using its resolution, too. The network is trained on the NLL loss

$$\sum_i -\log \mathcal{N}(z_i, w_{dsi}^{-2})(z_{i\,\mathrm{true}}), \tag{13}$$

where the true depth $z_{i\,\mathrm{true}}$ is obtained as $p_{i\,\mathrm{true}}^O$ above.

5 Pragmatic gPnP algorithm

Overall, the gPnP stage approximates the posterior distribution of T given (6) and optionally (9) as a Gaussian in pose space using (3). It is derived as follows:

5.1 Perspective measurements

We multiply (7) by $p^C_{i3}W_i$ making it linear in p^C_i with no C_i in the noise.

$$\eta'_i = W_i(p^C_{i3}(u_i - u_0) - f p^C_{i12}) \;=\; W_i \begin{pmatrix} -f & 0 & u_{i1}-u_{01} & 0 \\ 0 & -f & u_{i2}-u_{02} & 0 \end{pmatrix} p^C_i \tag{14}$$

$$:= W_i A_i p^C_i \;=\; W_i A_i T^C_R T p^O_i \qquad \eta'_i = p^C_{i3} W_i \eta_i \sim \mathcal{N}(0, (p^C_{i3})^2 I_2) \tag{15}$$

Gaussian measurements can only be fused if their relative covariance is known, however p^C_{i3}, the object point's camera-Z-coordinate, is unknown. Thus we treat the covariances as identical, *i.e.* $\eta'_i \overset{\text{approx}}{\sim} \mathcal{N}(0, \bar{z}^2 I_2)$, where $\bar{z}$ is the mean p^C_{i3}. This is a acceptable, since the distance camera to object is usually several times its size, so the p^C_{i3} differ only by a small factor (3% avg. on our data). The common factor $\bar{z}$ is still a-priori unknown, but will be recovered later in (25).

The term Tp^O_i is linear in T and hence can be expressed as a matrix-vector-product $\bar{p}^O_i \bar{T}$, if T is flattened as $\bar{T} \in \mathbb{R}^{13}$ and p^O_i converted into $\bar{p}^O_i \in \mathbb{R}^{4\times 13}$:

$$\bar{T} = \left(\begin{array}{ccccccccc|c|ccc} T_{11} & T_{12} & T_{13} & T_{21} & T_{22} & T_{23} & T_{31} & T_{32} & T_{33} & 1 & T_{13} & T_{23} & T_{33} \end{array} \right)^T, \tag{16}$$

$$\bar{p}^O_i = \left(\begin{array}{ccccccccc|c|ccc} p_1 & p_2 & p_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & p_1 & p_2 & p_3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_1 & p_2 & p_3 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right) \tag{17}$$

The flattened transform $\bar{T}$ consists of 9 rotations entries, followed by a constant 1 and 3 translation entries. Unlike [9,11,26] we directly estimate the transformation matrix instead of control points. The ordering of $\bar{T}$ will allow later to separate the translation as the 3-vector tail.

Now (15) becomes a linear measurement in $\bar{T}$ with i.i.d. Gaussian noise:

$$\eta'_i = W_i A_i T^C_R T p^O_i = (W_i A_i T^C_R \bar{p}^O_i)\bar{T} =: M_i \bar{T}. \tag{18}$$

Consider the column $M_{\bullet,10}$ which is multiplied by the fixed $\bar{T}_{10} = 1$. From the pattern in A_i and $\bar{p}^O_i$ it is zero, if and only if the translation in T^C_R is zero. This shows, that it is needed in general but not for a single camera ($T^C_R = I$).

Following the usual least-squares approach, the M_i are stacked as $M = (M_{1\ldots n})$ and assuming independence $M\bar{T} \sim \mathcal{N}(0, \bar{z}^2 I_{2n})$. The negative log-likelihood of $\bar{T}$ is now expressed by a $\mathbb{R}^{13\times 13}$ matrix $M^T M = \sum_i M_i^T M_i$ as

$$-\log p(T|u_{1\ldots n}, p^O_{1\ldots n}) + \text{const} = \bar{z}^2 \|M\bar{T}\|^2 = \bar{z}^2 \bar{T}^T \big(M^T M\big)\bar{T}, \tag{19}$$

and shall be minimized for $T \in SE(3)$. For several cameras, their M's are stacked, resp. their $M^T M$'s added. So far, the approach resembles [9] except, that we weight by W_i and estimate T directly instead of control points.

5.2 Depth measurements

The depth measurements z_i are handled similarly multiplying (10) with z_i:

$$\eta'_{di} = z_i w_{di}\big(z_i - p^C_{i3}\big) = \begin{pmatrix}0\,0 & -w_{di}z_i & w_{di}z_i^2\end{pmatrix} p_i^C =: B_i p_i^C \tag{20}$$

$$= (B_i T_R^C \bar{p}_i^O)\bar{T} =: M_{di}\bar{T}, \quad \eta'_{di} = z_i\eta_{di} \sim \mathcal{N}(0, z_i^2) \tag{21}$$

This makes also $\eta'_{di} \overset{\text{approx}}{\sim} \mathcal{N}(0, \bar{z}^2)$ so the M_{di} can be stacked with M_i or equivalently $\sum_i M_{di}^T M_{di}$ is added to M^TM. This fuses perspective and depth measurements into one matrix. Even for $T_R^C = I$ it requires the $M_{\bullet,10}$ column. Indeed, the fixed $\bar{T}_{10}$ entry is necessary whenever scale is defined even without the $T \in SE(3)$ costraint, because without $\bar{T}_{10}$ any linear equation in $\bar{T}$ is scale-ambiguous.

5.3 Gauss-Newton iteration

Following the approach from [3] specifically (5) to minimize (19) over T by Gauss-Newton we need to minimize $\|M\overline{\breve{T}\boxplus\delta}\|^2$ for δ in a linearized way in each iteration, where $\breve{T} = T_k$ is the starting point of the current iteration. We therefor write the linearization (1) for $\breve{T}\boxplus\delta$ as a matrix-vector product $P\binom{\delta}{1}$.

$$\overline{\breve{T}\boxplus\delta} \approx \begin{pmatrix} 0 & -\breve{T}_{i3} & +\breve{T}_{i2} & 0 & 0 & 0 & \breve{T}_{i1} \\ +\breve{T}_{i3} & 0 & -\breve{T}_{i1} & 0 & 0 & 0 & \breve{T}_{i2} \\ -\breve{T}_{i2} & +\breve{T}_{i1} & 0 & 0 & 0 & 0 & \breve{T}_{i3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \breve{T}_{i1} & \breve{T}_{i2} & \breve{T}_{i3} & \breve{T}_{i4} \end{pmatrix}\begin{matrix}\}\text{i=1..3}\\ \\ \\ \\ \}\text{i=1..3}\end{matrix} \begin{pmatrix}\delta\\1\end{pmatrix} =: P\begin{pmatrix}\delta\\1\end{pmatrix} \tag{22}$$

The first block of 3 rows is repeated for $i = 1\ldots3$ (rotation in row-wise ordering) and also the last block (translation). It results a 6-D quadratic minimization

$$\left\|M\overline{\breve{T}\boxplus\delta}\right\|^2 \approx \tbinom{\delta}{1}^T \big(P^TM^TMP\big)\tbinom{\delta}{1} =: \tbinom{\delta}{1}^T Q\tbinom{\delta}{1}, \tag{23}$$

$$\delta^* = \arg\min_\delta \tbinom{\delta}{1}^T Q\tbinom{\delta}{1} = -Q^{-1}_{1\ldots6,1\ldots6}Q_{1\ldots6,7}, \quad T_{k+1} = \breve{T}\boxplus\delta^* \tag{24}$$

and following (5), $T_{k+1} = \breve{T}\boxplus\delta^*$ is the result of one Gauss-Newton iteration.

In Gauss-Newton, the final inverse pseudo-Hessian $Q^{-1}_{1\ldots6,1\ldots6}$ is the covariance of the estimate [14], here in the sense of (3). However, there is still the unknown factor $\bar{z}^2$ from (19). In theory, the obtained minimum of (23) should be measurement dimension minus state dimension on average, *i.e.* $\text{nRow}(M)-6$. We use this to estimate a scaling factor for the covariance [14]. It gets rid of $\bar{z}^2$ and any wrong factor in the weights predicted by the CNN. Such a factor arises, because they are trained on *training* data and quantify the prediction error on *training* data which is lower than on test data (Sec.6, Fig. 6).

$$T_{\text{gPnP}} = \breve{T}\boxplus\delta^*, \quad \Sigma_{\text{gPnP}} = \frac{T_{\text{gPnP}}T^TM^TMT_{\text{gPnP}}}{\text{nRow}(M)-6}Q^{-1}_{1\ldots6,1\ldots6} \tag{25}$$

As a final remark: The Gauss-Newton iterations are very efficient, because they work on the fixed size M^TM instead of the original measurements. During training and for all tests we use ten iterations albeit empiric results indicate that four iterations might already be sufficient for full convergence.

5.4 Calculation of the initial guess

Iterative algorithms need an initial guess, however we are far from the minimal case that makes *e.g.* [9,11,26] so complex. So we decompose into a rotation $\bar{R} \in \mathbb{R}^{10}$ (with the fixed 1) and a translation $t \in \mathbb{R}^3$ solved by Schur-complement [14].

$$\min_{T \in SE(3)} \bar{T}^T M^T M \bar{T} = \min_{R \in SO(3)} \min_{t \in \mathbb{R}^3} \left(\begin{smallmatrix} \bar{R} \\ t \end{smallmatrix}\right)^T \begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix} \left(\begin{smallmatrix} \bar{R} \\ t \end{smallmatrix}\right) \tag{26}$$

$$= \min_{R \in SO(3)} \bar{R}^T \left(P - Q^T S^{-1} Q\right) \bar{R}. \approx \min_{R \in \mathrm{SO}_d} \bar{R}^T \left(P - Q^T S^{-1} Q\right) \tag{27}$$

$$t^*(\bar{R}) = \arg\min_{t \in \mathbb{R}^3} \left(\begin{smallmatrix} \bar{R} \\ t \end{smallmatrix}\right)^T \begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix} \left(\begin{smallmatrix} \bar{R} \\ t \end{smallmatrix}\right) = -QS^{-1}\bar{R} \tag{28}$$

For (27) we simply try 960 fixed rotations $R \in \mathrm{SO}_d$ precomputed by choosing the x-axis on the 60 vertices of a truncated icosahedron (football, $\approx 23°$ spaced), followed by an x-rotation in 22.5° steps. With (28) we exclude objects behind the camera ($t^*(\bar{R})_3 \leq 0$) and get the final translation.

6 Experimental Results

We evaluate our system on the YCB-Video dataset [23]. We exclude symmetric objects, because the network cannot be expected to distinguish symmetric points[1]. Handling this is future work. For training we use the data provided by [23] as well as additionally generated data according to the approach by [22].

We follow [23] and utilize the ADD-error as error metric, which is defined for a model with a finite set of object points M and two poses T, T' as

$$ADD(T, T') = \frac{1}{|M|} \sum_{p \in M} ||Tp - T'p||. \tag{29}$$

In addition, since we estimate uncertainty of the predicted pose, we also compute the expected ADD-error ($\widehat{ADD}$) from Σ and M by

$$\widehat{ADD} = \frac{1}{|M|} \sum_{p \in M} \hat{N}\left(\left([p]_\times, I_3\right) \Sigma \left([p]_\times, I_3\right)^T\right),\ a = \sqrt{\lambda_1/\lambda_3}, b = \sqrt{\lambda_2/\lambda_3},$$

$$\hat{N}(C) \approx \sqrt{\lambda_3}\left(0.782 + 0.234(a+b) + 0.246(a^2+b^2) - 0.126ab\right). \tag{30}$$

$\lambda_1 \leq \lambda_2 \leq \lambda_3$ are the eigenvalues of C. We cannot explain this approximation here for lack of space, but have verified it has 1.3% average (2.9% max) error.

Figure 4 shows object-wise accuracy-threshold curves for ADD-errors in the range of $[0\ m, 0.1\ m]$ both with and without depth-data. In addition to the actual errors we show the expected accuracy-threshold curves based on our predicted uncertainties, which are obtained as the ADD-error from samples from the posterior 6d error distribution $\mathcal{N}(0, \Sigma)$. One can see that the network's prediction

[1] This is, because the network is trained to distinguish each and every object point based on its appearance (which is the same for multiple points within a symmetrical object).

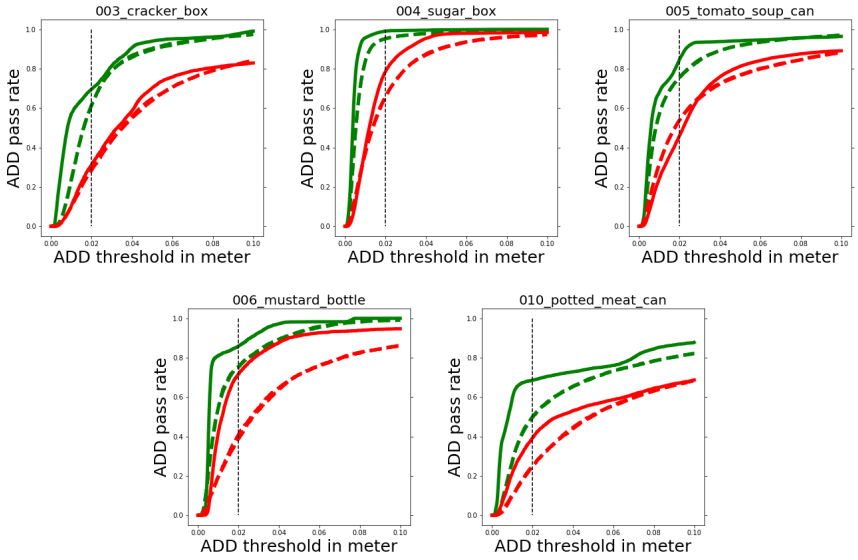


Fig. 4. Accuracy-Treshold curves for evaluated objects with RGB input (red solid lines) and with additional depth fusion (green solid lines). Dashed lines describe the expected accuracy based on the predicted uncertainties. The vertical line marks the two centimeter spot. Due to space we depict the object subset which was used in [22].

about its own accuracy curve looks fairly close to its actual accuracy curve, indicating that the uncertainty information Σ is meaningful. The curves also show that incorporating depth greatly improves the result.

Table 1 compares the performance PoseCNN [23] and DOPE [22]. It lists the area under these ADD-accuracy-treshold-curves for all objects. In addition to our proposed architecture, which utilizes the segmentation of PoseCNN (cf. 4), we also state comparative results using the true segmentation instead. The intention here is to evaluate the effect of the segmentation algorithm selection.

One can see that there are no big differences between either segmentation use. An exception makes the object 009_gelatin_box where the true segmentation performs much higher, because the segmentation has some outliers.

Our results fluctuate between different objects. In general our detection performance (without the uncertainty estimation) is the higher the better possible it is to distinguish all regions of an object. Otherwise, dissenting p_i^O regions might accrue, disturbing gPnP (which is strongest in case of 011_banana).

Table 1 also shows our results with depth data fusion compared to PoseCNN with iterative closest point (ICP). It can be clearly seen that our results benefit highly from the depth fusion but the more powerful ICP produces overall better results by the cost of much higher computation time. ICP may also dismiss known rotations of objects where the 3D-model (without its texture) is rotational invariant, which results in worse ADD-errors than our depth fusion.

YCB-Video dataset	RGB				RGB-D		
Method / Object	[23]	[22]	Seg_{gt}	Our	$[23]^{\text{ICP}}$	$\text{Seg}_{gt}^{\text{DF}}$	Our^{DF}
002_master_chef_can	50.9	-	56.3	**54.6**	69.0	78.0	**78.1**
003_cracker_box	51.7	55.9	57.7	**57.6**	80.7	80.9	**82.6**
004_sugar_box	68.6	75.7	85.5	**84.1**	**97.2**	95.8	95.3
005_tomato_soup_can	66.0	**76.1**	69.9	68.3	81.6	87.5	**86.6**
006_mustard_bottle	79.9	**81.9**	79.0	79.0	**97.0**	89.9	90.0
007_tuna_fish_can	**70.4**	-	42.6	43.5	83.1	84.5	**84.4**
008_pudding_box	**62.9**	-	51.0	50.3	**96.6**	91.4	92.0
009_gelatin_box	**75.2**	-	86.2	74.8	**98.2**	96.7	95.5
010_potted_meat_can	**59.6**	39.4	50.9	50.3	**83.8**	72.0	72.1
011_banana	**72.3**	-	8.6	8.2	**91.6**	45.6	45.8
019_pitcher_base	52.5	-	77.7	**77.8**	**96.7**	92.7	92.7
021_bleach_cleanser	50.5	-	60.4	**59.3**	**92.3**	83.1	82.7
025_mug	57.7	-	69.8	**69.1**	81.4	88.4	**91.0**
035_power_drill	55.1	-	71.5	**71.4**	**96.9**	89.3	88.3
AVG	62.4	(65.8)	61.9	60.6	88.3	84.0	84.1

Table 1. Pose evaluation: Area under ADD-accuracy-treshold-curve ([0,10] cm). `Our` method is compared against PoseCNN's [23] and DOPE's [22] results - RGB-only and with depth fusion ($^{\text{DF}}$) or respective ICP. $\texttt{Seg}_{gt}$ state comparative results using our's with true segmentation.

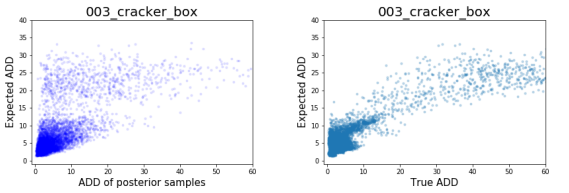


Fig. 5. Example expected ADD vs. posterior samples (left) and vs. true ADD-errors (right).

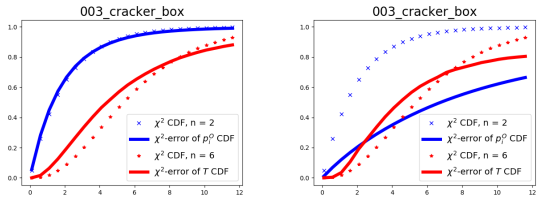
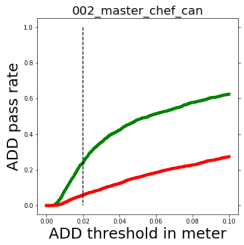


Fig. 6. Cummulative χ^2 curves on training (left) and test images (right). Solid lines show the measured curves for points and poses resp. The dotted lines show the theoretical curves, corresponding to perfectly correct posterior distributions.



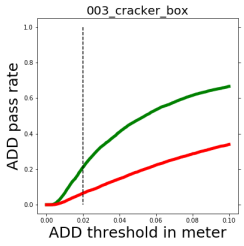


Fig. 7. Example Accuracy-Treshold curves for monocular (red) and binocular (green) results.

Figure 6 investigates the provided uncertainty information. The χ^2-error measures consistency between actual error and estimated uncertainty. For the training data, both points p_i^O and pose T uncertainties are good. For the test data, the CNN underestimates the uncertainty in p_i^O, naturally, since it is trained on training data, where the p_i^O are better. Still the gPnP computes a meaningful uncertainty from that, because it normalizes with the actual residuum in (25).

Fig.5 shows a distribution of actual vs. expected ADD (by (30)) and how that would look if the posterior was perfectly correct. Both match qualitatively, showing that the system mostly knows when it performs badly. Interestingly actual ADD is never small, when expected ADD is very large.

At last we want to show that our system is able to combine multiple cameras for one detection. Therefore we generated a small simulated binocular dataset, whereon we compare our monocular with our binocular results (Fig. 7). One can see that the binocular results are much higher.

7 Conclusion and Future Work

To our knowledge we are the first to present a 6d pose estimation CNN that predicts its own uncertainty (per estimate) with meaningful accuracy. Our network system predicts observed object points per pixel as well as per pixel their uncertainty in the image plane. This information is fused through a gPnP resulting in a pose estimate with covariance. Currently, our pose prediction (not the uncertainty estimation) can be confused by not clearly distinguishable object regions which we want to study in future work including also texture-less objects, rotation invariance and occlusion, as well as, train a single CNN for all objects.

Acknowledgments

The research reported in this paper has been supported by the German Research Foundation DFG, as part of Collaborative Research Center (Sonderforschungsbereich) 1320 EASE - Everyday Activity Science and Engineering, University of Bremen (`http://www.ease-crc.org/`). The research was conducted in subproject R02 Multi-cue perception supported by background knowledge.

References

1. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In: ECCV (2014)
2. Crivellaro, A., Rad, M., Verdie, Y., Moo Yi, K., Fua, P., Lepetit, V.: A novel representation of parts for accurate 3d object detection and tracking in monocular images. In: ICCV. pp. 4391–4399 (2015)
3. Hertzberg, C., Wagner, R., Frese, U., Schröder, L.: Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. Information Fusion **14**(1), 57–77 (2013)

4. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. pp. 4700–4708 (2017)
5. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In: ICCV (2017)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: NIPS. pp. 971–980 (2017)
8. Kneip, L., Furgale, P.: OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In: ICRA. pp. 1–8. IEEE (2014)
9. Kneip, L., Furgale, P., Siegwart, R.: Using multi-camera systems in robotics: Efficient solutions to the npnp problem. In: ICRA. pp. 3770–3776. IEEE (2013)
10. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
11. Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnp: An accurate O(n) solution to the pnp problem. IJCV **81**(2), 155 (2009)
12. Oberweger, M., Rad, M., Lepetit, V.: Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In: ECCV. pp. 119–134 (2018)
13. Periyasamy, A.S., Schwarz, M., Behnke, S.: Robust 6d object pose estimation in cluttered scenes using semantic segmentation and pose regression networks. In: IROS. pp. 6660–6666. IEEE (2018)
14. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C (2nd ed.): the art of scientific computing. Cambridge University Press, New York, NY, USA (1992)
15. Rad, M., Lepetit, V.: BB8: a scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In: ICCV. pp. 3828–3836 (2017)
16. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond (2018)
17. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: CVPR (2017)
18. Su, H., Qi, C.R., Li, Y., Guibas, L.J.: Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3d model views. In: ICCV (2015)
19. Sundermeyer, M., Marton, Z.C., Durner, M., Brucker, M., Triebel, R.: Implicit 3d orientation learning for 6d object detection from RGB images. In: ECCV. pp. 699–715 (2018)
20. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. In: CVPR. pp. 292–301 (2018)
21. Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics. MIT press (2005)
22. Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., Birchfield, S.: Deep object pose estimation for semantic robotic grasping of household objects. arXiv preprint arXiv:1809.10790 (2018)
23. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199 (2017)
24. Zakharov, S., Shugurov, I., Ilic, S.: DPOD: Dense 6D pose object detector in RGB images. arXiv preprint arXiv:1902.11020 (2019)
25. Zeng, A., Yu, K.T., Song, S., Suo, D., Walker, E., Rodriguez, A., Xiao, J.: Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In: ICRA. pp. 1386–1383. IEEE (2017)
26. Zheng, Y., Kuang, Y., Sugimoto, S., Astrom, K., Okutomi, M.: Revisiting the pnp problem: A fast, general and optimal solution. In: ICCV. pp. 2344–2351 (2013)