Learning Analysis-by-Synthesis for 6D Pose Estimation in RGB-D Images

Alexander Krull,* Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, Carsten Rother

TU Dresden
Dresden, Germany
*alexander.krull@tu-dresden.de

Abstract

Analysis-by-synthesis has been a successful approach for many tasks in computer vision, such as 6D pose estimation of an object in an RGB-D image which is the topic of this work. The idea is to compare the observation with the output of a forward process, such as a rendered image of the object of interest in a particular pose. Due to occlusion or complicated sensor noise, it can be difficult to perform this comparison in a meaningful way. We propose an approach that "learns to compare", while taking these difficulties into account. This is done by describing the posterior density of a particular object pose with a convolutional neural network (CNN) that compares observed and rendered images. The network is trained with the maximum likelihood paradigm. We observe empirically that the CNN does not specialize to the geometry or appearance of specific objects. It can be used with objects of vastly different shapes and appearances, and in different backgrounds. Compared to state-of-the-art, we demonstrate a significant improvement on two different datasets which include a total of eleven objects, cluttered background, and heavy occlusion.

1. Introduction

Tremendous effort has focused on the tasks of object instance detection and pose estimation in images and videos. In this paper, we consider pose estimation in a single RGB-D image, as shown in Fig. 1. Given the extra depth channel, it becomes feasible to extract the full 6D pose (3D rotation and 3D translation) of object instances present in the scene. Pose estimation has important applications in many areas, such as robotics [21, 33], medical imaging [24], and augmented reality [12]. Recently, Brachmann *et al*. [5] achieved state-of-the-art results by adapting an analysis-by-synthesis approach for pose estimation in RGB-D images. They use a random forest [6] to obtain pixelwise dense predictions. Building upon the system of [5], we propose a novel method to learn to compare in the analysis-by-synthesis framework. We use a convolutional neural network (CNN) inside a probabilistic context to achieve this.

Figure 1. Three pose estimation results from the *occlusion dataset* from [5] and [14]. Arrows indicate the positions of estimated and ground truth poses. The green silhouette indicates the ground truth pose, the blue silhouette corresponds to our estimated pose. Red indicates the pose estimate from [5]. The marker board served only for ground truth annotation.

Analysis-by-synthesis has been a successful approach for many tasks in computer vision, such as object recognition [13], scene parsing [15], pose estimation, and tracking [9]. A forward synthesis model generates images from pos-

sible geometric interpretations of the world, and then selects the interpretation that best agrees with the measured visual evidence. In particular for pose estimation, the idea is to compare the observation with the output of a forward process, such as a rendered image of the object of interest in a particular pose. When attempting pose estimation in RGB-D images, this comparison is non-trivial due to occlusion or complicated sensor noise. There are for example areas with no depth measurements in Kinect images due to poor IR-reflectance.

1.1. Contributions

- We achieve considerable improvements over state-of-the-art methods of pose estimation in RGB-D images with heavy occlusion.
- To the best of our knowledge, this work is the first to utilize a convolutional neural network (CNN) as a probabilistic model to learn to compare rendered and observed images.
- We observe that the CNN does not specialize to the geometry or appearance of specific objects, and it can be used with objects of vastly different shapes and appearances, and in different backgrounds.

The paper is organized as follows. Sec. 2 provides an overview of related work. Our proposed approach is described in Sec. 3. In Sec. 4 we present evaluation of our method compared to the state-of-the-art on two datasets. We conclude the paper in Sec. 5.

2. Related Work

A large body of work in computer vision has focused on the problem of object detection and pose estimation, including instance and category recognition, rigid and articulated objects, and coarse (quantized) and accurate (6D) poses. Pose estimation has been an active topic, ranging from template-based approaches [14, 8] over sparse feature-based approaches [21] to dense approaches [25, 5]. In the brief review below, we focus on techniques that specifically address CNNs and analysis-by-synthesis.

CNNs are driving advances in computer vision in areas such as image classification [16], detection [32], recognition [2, 23], semantic segmentation [20], and pose estimation [27]. CNNs have shown remarkable performance in the large-scale visual recognition challenge (ILSVRC2012). The success of CNNs is attributed to their ability to learn rich feature representations as opposed to hand-designed features used in previous image classification methods. In [11], rich image and depth feature representations have been learned with CNNs to detect objects in RGB-D images. In [1], CNNs are used to generate an RGB image given the set of 3D chair models, the chair type, viewpoint and color. Very recent work from Gupta *et al*. [10] uses object instance segmentation output from [11] to infer the 3D object pose in RGB-D images. Another CNN is used to predict the coarse pose of the object. This CNN is trained using pixel normals in images containing rendered synthetic objects. This coarse pose is used to align a small number of prototypical models to the data, and place the model that fits the best into the scene. In contrast to the above approaches, we use a CNN to compare rendered and observed images. The output of our CNN is a single energy value, while in [10] the output of the CNN is the object pose. In [7], a similarity metric is learned. The learning process minimizes a discriminative loss function. A CNN with *siamese* architecture is used to map two faces to a feature space for comparison. Similarly, in [30] Wohlhart and Lepetit train a CNN to map image patches to a descriptor space, where pose estimation and object recognition is solved using the nearest neighbor method. Our framework is probabilistic. The posterior distribution of the pose is modelled as a Gibbs distribution with a CNN as energy function. Zbontar and LeCun [31] train a CNN to predict how well two image patches match and use it to compute the stereo matching cost. The cost is minimized by cross-based cost aggregation and semi-global matching, followed by a left-right consistency check to eliminate errors in occluded regions. While in [31] the CNN is used for comparing two image patches, our CNN is used to compare rendered and observed images.

Analysis-by-synthesis has been a successful approach for many tasks in computer vision, such as object recognition [13], scene parsing [15], viewpoint synthesis [13], material classification [29], and gaze estimation [26]. All these approaches use a forward model to synthesize some form of image, which is compared to observations. Many works learn a feature representation and compare in feature space. For instance, in [13] the analysis-by-synthesis strategy has been used for recognizing and reconstructing 3D objects in images. The forward model synthesizes visual templates defined on invariant features. In [28] differentiable features are used to facilitate optimization. Gall *et al*. [9] propose an analysis-by-synthesis framework for motion capture and tracking. It combines patch-based and region-based matching to track body parts. Patch-based matching extracts correspondences between two successive frames for prediction as well as between the current image and a synthesized image to avoid drift. Recently, Brachmann *et al*. [5] achieved state-of-the-art results by adapting a classical analysis-by-synthesis approach for 6D pose estimation of specific objects from a single RGB-D image. They use a new representation in form of a joint dense 3D object coordinate and object class labeling. The major difference to our work is that we learn to compare in the analysis-by-synthesis approach. For the problem of 6D pose estimation, due to occlusion or complicated sensor noise, it can be difficult to compare the

observation with the output of a rendered image of the object of interest in a particular pose. In this paper, we propose an approach, which draws on recent successes of CNNs. Different from aforementioned approaches, we model the posterior density of a particular object pose with a CNN that compares an observed and rendered image. The network is trained with the maximum likelihood paradigm. One of the most closely related works is [18]. They use a CNN as a part of probabilistic model. The CNN is fed in a sequential manner, first with the rendered image, then with the observed image. This produces two feature vectors, which are compared in the subsequent step, to give the probability of the observed image. In contrast to [18], we jointly input the rendered and observed images into a CNN to produce an energy value. The major difference is that our CNN is trained, while they use a pre-trained CNN as feature extractor.

2.1. Review of the Pose Estimation Method [5]

We will now describe the system from [5] in detail, because it is of particular relevance for our method. Brachmann *et al*. [5] achieved state-of-the-art results by using a random forest [6] to obtain pixelwise dense predictions, which facilitate pose estimation. Each tree in their forest is trained to jointly predict to which object a pixel belongs, and where it is located on the surface of this object. A tree outputs a soft segmentation image for each object with values between 0 and 1, indicating whether a pixel belongs to the object or not. The predictions of different trees are then combined to a single object probability. Additionally, each tree outputs 3D object coordinates for each object and each pixel. The term object coordinates refers to the coordinates in the local coordinate system of the object. When estimating the pose of a particular object, Brachmann *et al*. [5] utilize the forest predictions in two ways:

Firstly, it is used to define an *energy function*, which is minimized to obtain the final pose. All aspects of the energy follow the analysis-by-synthesis principle. It is based on a pixelwise comparison between the predictions, the recorded depth values and rendered images of the object in the particular pose. In detail, three comparisons are done: (a) the rendered depth image of the object is compared to the recorded depth image; (b) the rendered image of object coordinates is compared to the predicted object coordinates; (c) the rendered segmentation mask of the object is compared to the predicted object class probability for the object. The pixelwise error inside the segmentation mask is aggregated and divided by the area of the mask. Robust error measures are used to deal with outliers.

Secondly, they use the forest predictions for an efficient *optimization scheme* to minimize the energy described above. It consists of two steps. The pixelwise object class probabilities are used inside the RANSAC pose estimation. In detail, sets of three pixels are sampled depending on the object class probability. For each set a pose hypothesis is calculated using the 3D-3D-correspondences between the camera coordinates, provided by the depth camera, and the object coordinates predicted by the forest. The best hypotheses, according to the energy function, are refined in a final step. Refinement is done by repeatedly determining inlier pixels in the rendered mask of the object, and again using the correspondences they provide to calculate a better pose. Finally. the pose with the lowest energy is taken as the final estimate.

In our work, we build upon the framework of [5]. As in [5] we use the regression-classification random forest to obtain the predictions described above. We also use their optimization scheme, but replace the energy function with a novel one, based on a CNN, that is trained. The key difference is that while the energy function in [5] has only a few parameters which can be trained via discriminative cross-validation, the CNN has around 600K, which we train with a maximum likelihood objective. We show that this richness of parameters makes remarkable difference, and practical challenges such as occlusion and noise are much better dealt with. This approach will be described in the next section.

3. Method

We will first give a description of the pose estimation task and introduce our terminology. Then we will describe our probabilistic model. The heart of this model is a CNN, which will be discussed subsequently. This is followed by a description of our maximum likelihood training procedure of the probabilistic model. Finally, our inference procedure at test time is described. Fig. 2 gives an overview of our energy evaluation pipeline.

3.1. The Pose Estimation Task

We will now formally define the task of 6D pose estimation. Our goal is to estimate the pose H of a rigid object[1] from a set of observations denoted by $\mathbf{x}$, which will be discussed later. A pose describes the transformation from the local coordinate system of the object to the coordinate system of the camera. The local coordinate system has its origin in the center of the object. Each pose $H = (R, T)$ is a combination of two components. The rotational component R is a 3×3 matrix describing the rotation around the center of the object. The translational component T is a 3D vector corresponding to the position of the object center in the camera coordinate system.

Let us now describe the observation $\mathbf{x}$ that is used to estimate the object pose. We use RGB-D images as input. However, since we use the same random forest predictions

[1] It should be noted that we assume the object to be present in the field of view, *i.e*. we do not perform object recognition.

as in [5], the term observation or observed images will refer to two parts: (a) the forest predictions as described in [5], as well as (b) the recorded depth image. The reason for this simplified view is that the focus of our work lies on the modeling of the posterior density and aspects of the random forest prediction.

3.2. Probabilistic Model

We model the posterior distribution of the pose H given the observations $\mathbf{x}$ as a Gibbs distribution

$$p(H|\mathbf{x};\boldsymbol{\theta}) = \frac{\exp\big(-E(H,\mathbf{x};\boldsymbol{\theta})\big)}{\int \exp\big(-E(\hat{H},\mathbf{x};\boldsymbol{\theta})\big)d\hat{H}}\,, \qquad (1)$$

where $E\,(H,\mathbf{x};\boldsymbol{\theta})$ is the so called energy function. The energy function is a mapping from a pose H and the observed images $\mathbf{x}$ to a real number, parametrized by the vector $\boldsymbol{\theta}$. Note that using a Gibbs distribution to model the posterior is a common practice for conditional random fields (CRFs) [19]. However, the underlying energies are quite different. While in a CRF the energy function is a sum of potential functions, we implement it by using a CNN which directly outputs the energy value. The parameter vector $\boldsymbol{\theta}$ holds the weights of our CNN.

3.3. Convolutional Neural Network

In order to implement the mapping from a pose H and the observed images $\mathbf{x}$ to an energy value we first render the object in pose H to obtain rendered images $\mathbf{r}(H)$. Our CNN then compares $\mathbf{x}$ with $\mathbf{r}(H)$ and outputs a value $f\big(\mathbf{x},\mathbf{r}(H);\boldsymbol{\theta}\big)$. We define the energy function as

$$E(H,\mathbf{x};\boldsymbol{\theta}) = f\big(\mathbf{x},\mathbf{r}(H);\boldsymbol{\theta}\big). \qquad (2)$$

Our network is trained to assign a low energy value when there is a large agreement between observed images and renderings and a high energy value when there is little agreement. To perform the comparison we use a simple architecture, in which we feed all rendered and observed images as separate input channels into the CNN.

Note that we consider only a square window around the center of the object with pose H. The width of the window is adjusted according to the size and distance of the object, as suggested by [5]. For performance reasons windows which are bigger than 100×100 pixels are down sampled to this size. We use in total six input channels for our network. Note that Fig. 2 shows the images from which these six input channels are derived.

One **observed depth** channel and one **rendered depth** channel that contain values in millimeters. They are normalized by subtracting the z component of the object position according to H.

One **rendered mask** channel of the object. Pixel values are either $+1$ for all pixels belonging to the object or -1 otherwise.

One **depth mask** channel indicating whether a depth value was measured in the pixel. Again, pixel values are either $+1$ for all pixels where a depth was measured or -1 otherwise.

One **probability** channel holding the combined pixel wise object probabilities from all trees. The values are re-scaled to lie between -1 and $+1$.

One **object coordinate** channel holding the pixel wise Euclidean distances between the rendered object coordinates and the predicted object coordinates from the tree giving the highest object probability for the respective pixel. We divide all values by the object diameter for normalization.

The $\tanh$ activation function is used after every convolution layer and after every fully connected layer. The first convolution layer C_1 consists 128 convolution kernels of size $3 \times 3 \times 6$. The second convolution layer C_2 consists of 128 kernels of size $3 \times 3 \times 128$, which is followed by a 2×2 max-pooling layer with stride 2 in each direction. The third convolution layer C_3 is identical to C_2. The fourth convolution layer C_4 consists of 256 kernels of size $3 \times 3 \times 128$. It is followed by a max-pooling operation over the remaining image size. The 256 channels are further processed by two fully connected layers with 256 neurons each and finally forwarded to a single output unit.

3.4. Maximum Likelihood Training

In training we want to find an optimal set of parameters $\boldsymbol{\theta}^*$ based on labeled training data $L = (\mathbf{x}_1, H_1^*)\dots(\mathbf{x}_n, H_n^*)$, where $\mathbf{x}_i$ shall denote observations of the i-th training image and H_i^* the corresponding ground truth pose. We apply the maximum likelihood paradigm and define

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{n} \ln p(H_i^*|\mathbf{x}_i;\boldsymbol{\theta}). \qquad (3)$$

In order to solve this optimization task we use stochastic gradient descent [3], which requires calculating the partial derivatives of the log likelihood for each training sample

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\ln p(H_i^*|\mathbf{x}_i;\boldsymbol{\theta}) = &-\frac{\partial}{\partial\theta_j}E\,(H_i^*,\mathbf{x}_i;\boldsymbol{\theta}) \\ &+\mathbb{E}\left[\frac{\partial}{\partial\theta_j}E\,(H,\mathbf{x}_i;\boldsymbol{\theta})\,\middle|\,\mathbf{x}_i;\boldsymbol{\theta}\right]\end{aligned} \qquad (4)$$

with respect to each parameter θ_j. Here $\mathbb{E}[\cdot|\mathbf{x}_i;\boldsymbol{\theta}]$ stands for the conditional expected value according to the posterior distribution $p(H|\mathbf{x}_i;\boldsymbol{\theta})$, parametrized by $\boldsymbol{\theta}$. While the partial derivatives of the energy function can be calculated by applying back propagation in our CNN, the expected value cannot be found in closed form. Therefore, we use the Metropolis algorithm [22] to approximate it, as discussed next.

Sampling. It is possible to approximate the expected value

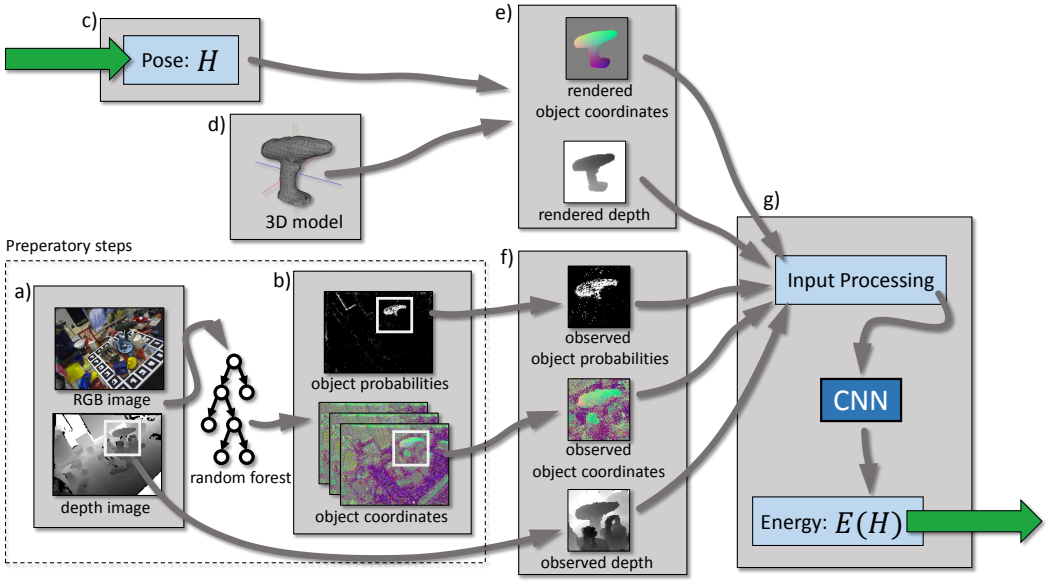

Figure 2. Our pipeline for the calculation of the energy function: Input and output are indicated by green arrows. The contents of the dashed box consists of preparatory steps, that have to be computed only once per image. (a) The RGB-D image we will base our estimate on. The image is processed by a random forest to calculate predictions. (b) The predicted object probabilities and object coordinates. In the probability image bright pixels indicate a high probability. In the object coordinate images the 3D object coordinates are mapped to the RGB cube for visualization. There are multiple object coordinate images. Each one represents the prediction of one tree. The object probabilities are combined to a single image [5]. (c) The pose we want to calculate the energy for. (d) A 3D model of the object. (e) Images produced by rendering the 3D model in the input pose. We render an object coordinate image and a depth image. We only use cutouts around the object. (f) Images of equal size are cut out from the predicted object probabilities, object coordinates and from the recorded depth image. (g) Finally, the rendered and observed images are processed and fed into the CNN (Sec. 3.3). The single output of the CNN is our energy function.

in Eq. (4) by a set of pose samples

$$\mathbb{E}\left[\frac{\partial}{\partial\theta_j}E\left(H,\mathbf{x}_i;\boldsymbol{\theta}\right)\middle|\mathbf{x}_i;\boldsymbol{\theta}\right] \approx \frac{1}{N}\sum_{k=1}^{N}\frac{\partial}{\partial\theta_j}E\left(H_k,\mathbf{x}_i;\boldsymbol{\theta}\right), \tag{5}$$

where $H_1 \dots H_N$ are pose-samples drawn independently from the posterior $p(H|\mathbf{x}_i;\boldsymbol{\theta})$ with the current parameters $\boldsymbol{\theta}$. We use the Metropolis algorithm [22] to generate these samples. It allows sampling from a distribution with a known density function that can be evaluated up to a constant factor. The algorithm generates a sequence of samples H_t by repeating two steps:

1. Draw a new proposed sample H' according to a proposal distribution $Q(H'|H_t)$.
2. Accept or reject the proposed sample according to an acceptance probability $A(H'|H_t)$. If the proposed sample is accepted set $H_{t+1} = H'$. If it is rejected set $H_{t+1} = H_t$.

The proposal distribution $Q(H'|H_t)$ has to be symmetric, *i.e.* $Q(H'|H_t) = Q(H_t|H')$. Our particular proposal distribution will be described in detail in the next paragraph. The acceptance probability is in our case defined as

$$A(H'|H_t) = \min\left(1, \frac{p(H'|\mathbf{x};\boldsymbol{\theta})}{p(H_t|\mathbf{x};\boldsymbol{\theta})}\right), \tag{6}$$

meaning that whenever the posterior density $p(H'|\mathbf{x};\boldsymbol{\theta})$ at the proposed sample is greater than the posterior density $p(H_t|\mathbf{x};\boldsymbol{\theta})$ at the current sample, the proposed sample will automatically be accepted. If this is not the case it will be accepted with the probability $p(H'|\mathbf{x};\boldsymbol{\theta})/p(H_t|\mathbf{x};\boldsymbol{\theta})$.

Proposal Distribution. A common choice for the proposal distribution is a normal distribution centered at the current sample. In our case this is not possible because the rotational component of the pose lives on the manifold $SO(3)$, *i.e.* the group of rotations. We define $Q(H'|H_t)$ implicitly by describing a sampling procedure and ensuring that it is symmetric. The translational component T' of the proposed sample is directly drawn from a 3D isotropic normal distribution $\mathcal{N}(T_t, \Sigma_T)$ centered at the translational component T_t of the current sample H_t. The rotational component R' of the proposed sample H' is generated by applying a random rotation $\hat{R}$ to the rotational component R_t of the current sample: $R' = \hat{R}R_t$.

We calculate $\hat{R}$ as the rotation matrix corresponding to

an Euler vector $\mathbf{e}$, which is drawn from a 3D zero centered isotropic normal distribution $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \Sigma_R)$.

Initialization and Burn-in-phase. When the Metropolis algorithm is initialized in an area with low density it requires more iterations to provide a fair approximation of the expected value. To find a good initialization we run our inference procedure (described in the next section) using the current parameter set. We then perform the Metropolis algorithm for a total of 130 iterations, disregarding the samples from the first 30 iterations which are considered as burn-in-phase.

3.5. Inference Procedure

During test time, we aim at finding the MAP estimate, *i.e*. the pose maximizing our posterior density as given in Eq. (1). Since the denominator in Eq. (1) is constant for any given observation $\mathbf{x}$, finding the MAP estimate is equivalent to minimizing our energy function. To achieve this, we utilize the optimization scheme from [5], but replace their energy function with ours.

4. Experiments

In the following, we compare our approach to the state-of-the-art method of Brachmann *et al*. in [5] for two different datasets. We first introduce the datasets. After that we describe details of our training procedure, and finally present quantitative and qualitative comparisons. We will see that we achieve considerable improvements for both datasets. Additionally, we observe that our CNN generalizes from a single training object to a set of 11 test objects, with large variability in appearance and geometry.

4.1. Datasets, Competitors, Evaluation Protocol

Datasets. We use two datasets featuring heavy occlusion. The first dataset was created by Brachmann *et al*. [5] by annotating the ground truth poses for eight partially occluded objects in images taken from the dataset of Hinterstoisser *et al*. [14]. We will refer to this dataset as the *occlusion dataset* from [5] and [14]. It includes a total of 8992 test cases (1214 images with multiple objects), which are used for testing. We choose this dataset because it is more challenging than the original dataset from [14], on which [5] already achieves an average of 98.3% correctly estimated poses.

The second dataset was introduced by Krull *et al*. in [17]. It provides six annotated RGB-D sequences of three different objects and consists of a total of 3187 images. We use three of the sequences for training and the other three (a total of 1715 test images) for testing.

Evaluation Protocol. We use the evaluation procedure as described in [5]. This means we calculate the percentage of correctly predicted poses for each sequence. As in [14], we calculate the average distance between the 3D model vertices under the estimated pose and under the ground truth pose. A pose is considered correct when the average distance is below 10% of the object diameter.

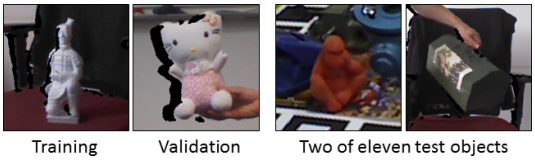

Figure 3. Images from one of our training-testing configurations: the *Samurai_1* sequence is used for training, the *Cat_1* for validation. Sequences of all objects are used for testing. Note, the objects are of vastly different shape and appearance.

Competitors. We compare our method to the one presented in [5]. For doing so we needed to re-implement this method. We observed that our re-implementation gives on average slightly superior results. In the following, we mostly report two numbers, those of our re-implementation and those of the method of [5], reported in [5] or [17]. For completeness, we additionally provide the numbers from LineMOD [14] as reported in [5].

4.2. Training Procedure

Random Forests. We used different random forests for training and testing on both datasets. The forests were kindly provided to us by the authors of [5].

CNN. We trained three CNNs, each time using only a single object from the dataset provided by Krull *et al*. in [17]. The sequences *Toolbox_1*, *Cat_1*, and *Samurai_1* served as training sets - see Fig. 3. The first 100 frames from *Samurai_1* were removed in order to obtain a high percentage of frames with occlusion. Our validation set consists of 100 randomly selected frames from the *Cat_1* sequence, or the *Samurai_1* sequence (in the case where *Cat_1* was used as training set). The weights of the CNN were randomly initialized. Before training, the random weights of the last layer were multiplied by factor 1000, in order to cover a greater range of possible energy values. After every 5th iteration of stochastic gradient descent, we perform inference on the validation set and adjust the learning rate. The learning rate at step t was proportional to $\gamma_t = \gamma_0/(1 + \lambda t)$ [4], with $\gamma_0 = 10$ and $\lambda = 0.5$. After training we pick the set of weights which achieved the highest percentage of correctly estimates poses on the validation set. We use the criterion from [14] to classify a pose as correct. One training cycle consisting of five steps of stochastic gradient descent and validation took[2] 9min 46sec (2min 27sec + 7min 19sec). Further details on our training procedure can be found in

[2]We used an *Intel(R) Core (TM) i7-3820 CPU at 3.60GHz* with GeForce GTX 660 GPU. The *Cat_1* sequence was used for training and 100 random frames from *Samurai_1* for validation.

the supplementary material.

4.3. Comparison

***Occlusion Dataset* from [5] and [14].** Quantitative results for this dataset are shown in Fig. 4, for all individual test and training objects. Considering the average over all objects we achieve an improvement of up to 9.23% compared to our re-implementation of [5] and **10.4%** compared to the reported values in [5]. Some qualitative results are illustrated in Fig. 7. In Fig. 5 we show another comparison of our method with respect to [5]. It illustrates that we achieve the biggest gain for occlusions between 50% and 60%.

Dataset of Krull *et al.* For this dataset we observe similar results as with the previous dataset. Since the other sequences were used in training and validation, we evaluated only with the *Toolbox_2*, *Cat_2*, and *Samurai_2* sequences. When averaged over all objects we achieve an improvement of **10.97%** compared to the results of [5]. The quantitative results can be found in Fig. 6, and a few qualitative results are shown in Fig. 8.

Discussion of Failure Cases. The failure cases which are framed red in Fig. 7 have to be considered as failure of our learned energy function. However, the failure cases framed orange still exhibit a lower energy at the ground truth pose than at the estimate. This indicates a failure of the optimization scheme. It should be investigated in which case the correct pose can be found using an alternative optimization scheme. In the dataset introduced by Krull *et al.* our accuracy for the *Tool Box* sequences is below the one of our competitor (see Fig. 6). We attribute this to the fact that the *Tool Box* is the biggest object and most strongly affected by the down sampling schema described in Sec. 3.3.

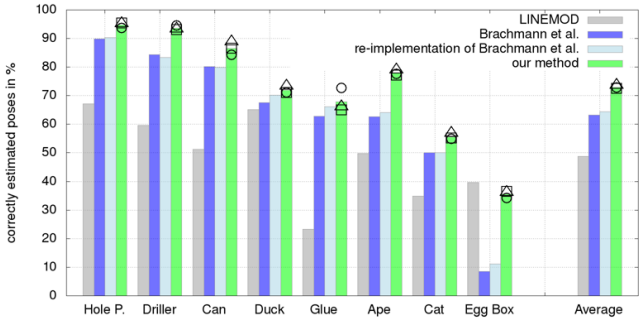

Figure 4. Quantitative comparison of our method against the results of [5] and LineMOD [14] on the *occlusion dataset* from [5] and [14]. *Circles*, *Squares*, and *Triangles* indicate the individual performance of CNNs trained with *Tool Box*, *Cat*, and *Samurai* respectively. The green bars indicate the average result. Averaged over all test and training objects we obtain the correct pose in **72.98%** of cases, in contrast to 63.24% for [5] and 48.84% for LineMOD [14]. A table with the the detailed numbers can be found in the supplementary material.

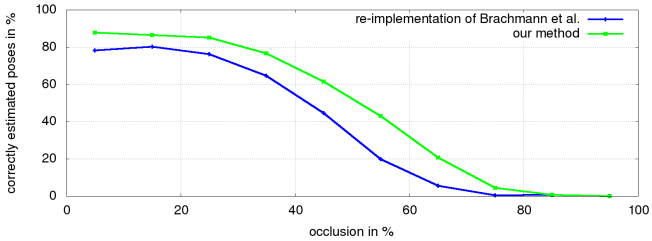

Figure 5. The percentage of correctly estimated poses for all test cases of the *occlusion dataset* from [5] and [14], as a function of the level of occlusion. For this we divided the test cases into bins according to the amount of occlusion, using a bin width of 10%. (See details of this procedure in the supplementary material.) We compare our method (using the CNN trained with the *Samurai* object) to our re-implementation of [5]. We achieve improvements of over 20% for occlusion levels between 50% and 60%.

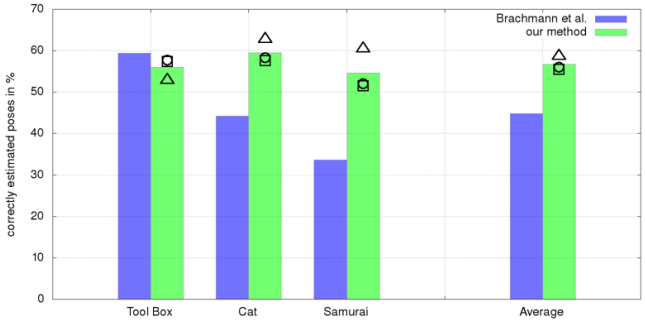

Figure 6. Comparison of our method on the dataset of Krull *et al.*, against the results of [5]. *Circles*, *Squares*, and *Triangles* indicate the individual performance of CNNs trained with *Tool Box*, *Cat*, and *Samurai* respectively. The green bars indicate the average result. We report 56.02%, 59.56%, and 54.65% correctly estimated poses for *Tool Box*, *Cat*, and *Samurai* respectively. Averaged over all test and training objects we achieve **56.74%**.

5. Conclusion

We have presented a model for the posterior distribution in 6D pose estimation, which uses a CNN to map rendered and observed images to an energy value. We train the CNN based on the maximum likelihood paradigm. It has been demonstrated that training on a single object is sufficient and the CNN is able to generalize to different objects and backgrounds. Our system has been evaluated on two datasets featuring heavy occlusion. By using our energy as objective function for pose estimation, we were able to achieve considerable improvements compared to the best previously published results.

Our approach is not restricted to the feature channels and even the application we demonstrated. The architecture can in principle be applied to any kind of observed and rendered image. We think it would be worth investigating if the ap-

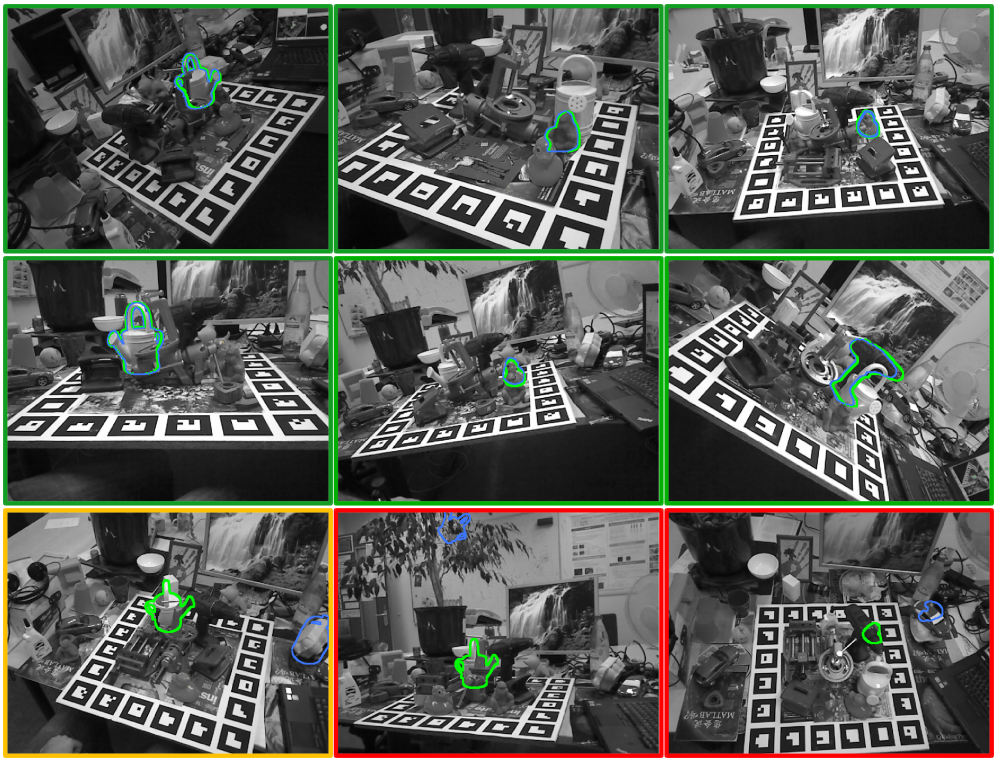

Figure 7. Qualitative results of our method on the *occlusion dataset* from [5] and [14]. Here, green and blue silhouettes correspond to the ground truth and our estimate, respectively. The test images depicted with a green frame show correct estimates. Images with orange and red frame show incorrect estimates. The image with an orange frame shows a case where the energy of the ground truth pose, according to Eq. (2), is lower than the energy of the estimated pose. In this case a better pose may be found with an improved optimization scheme.

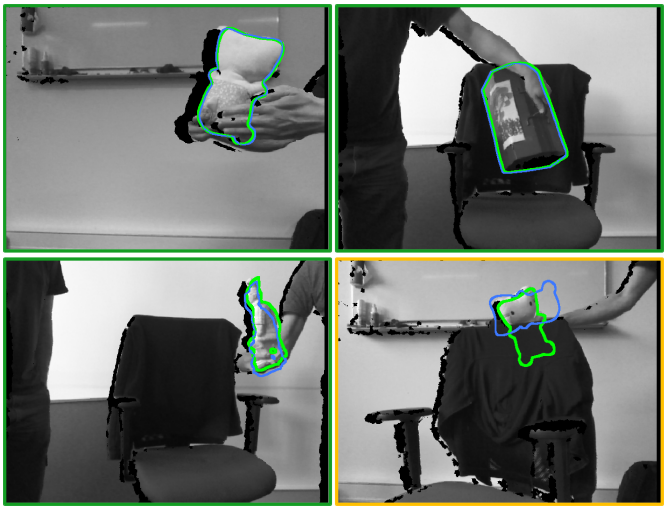

Figure 8. Qualitative results of our method on the test cases from the dataset introduced in [17]: Green frames correspond to correctly estimated poses according to the criteria from [14]. Orange frames correspond to incorrectly estimated poses with a lower energy at the ground truth than at the estimated pose.

proach could be applied to other scenarios. An example could be pose estimation from pure RGB without recorded depth image and a forest to calculate features. Pose estimation for object classes could also benefit from our approach. Considering the recent success of CNNs in recognition [2, 23] it might be possible for a CNN to learn to compare observed images to renderings of an idealized model representing an object class instead of an instance. Our approach is not limited to comparing images of the same kind, as for example rendered and observed depth images. Instead, it could learn to asses the plausibility of the shading in an observed RGB image by comparing it to a rendered depth image, which can be more easily produced than a realistic RGB rendering.

An interesting future line of research could be to train a CNN to predict pose updates from observed and rendered images. This could replace the refinement step and might improve the results.

Acknowledgements: This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 647769).

References

[1] A.Dosovitskiy, J.T.Springenberg, and T.Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015. 2

[2] P. Agrawal, R. B. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, pages 329–344, 2014. 2, 8

[3] L. Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8), 1991. 4

[4] L. Bottou. Stochastic gradient tricks. In *Neural Networks, Tricks of the Trade, Reloaded*, pages 430–445. 2012. 6

[5] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, pages 536–551, 2014. 1, 2, 3, 4, 5, 6, 7, 8

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 1, 3

[7] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, pages 539–546, 2005. 2

[8] M. Dantone, J. Gall, C. Leistner, and L. J. V. Gool. Body parts dependent joint regressors for human pose estimation in still images. *PAMI*, 36(11):2131–2143, 2014. 2

[9] J. Gall, B. Rosenhahn, and H. Seidel. Drift-free tracking of rigid and articulated objects. In *CVPR*, 2008. 1, 2

[10] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Inferring 3d object pose in RGB-D images. In *CVPR*, 2015. 2

[11] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, pages 345–360, 2014. 2

[12] N. Hagbi, O. Bergig, J. El-Sana, and M. Billinghurst. Shape recognition and pose estimation for mobile augmented reality. *IEEE Trans. Vis. Comput. Graph.*, 17(10):1369–1379, 2011. 1

[13] M. Hejrati and D. Ramanan. Analysis by synthesis: 3d object recognition by object reconstruction. In *CVPR*, pages 2449–2456, 2014. 1, 2

[14] 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. 1, 2, 6, 7, 8

[15] P. Isola and C. Liu. Scene collaging: Analysis and synthesis of natural images with semantic layers. In *ICCV*, pages 3048–3055, 2013. 1, 2

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 2

[17] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother. 6-dof model based tracking via object coordinate regression. In *ACCV*, 2014. 6, 8

[18] T. Kulkarni, I. Yildirim, W. Freiwald, and J. Tenenbaum. Deep generative vision as approximate bayesian computation. In *NIPS ABC Workshop*, 2014. 3

[19] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001. 4

[20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[21] M. Martinez Torres, A. Collet Romea, and S. Srinivasa. Moped: A scalable and low latency object recognition and pose estimation system. In *ICRA*, 2010. 1, 2

[22] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953. 4, 5

[23] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724, 2014. 2, 8

[24] K. Ralovich, M. John, E. Camus, N. Navab, and T. Heimann. 6dof catheter detection, application to intracardiac echocardiography. In *MICCAI*, 2014. 1

[25] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. W. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, pages 2930–2937, 2013. 2

[26] Y. Sugano, Y. Matsushita, and Y. Sato. Learning-by-synthesis for appearance-based 3d gaze estimation. In *CVPR*, pages 1821–1828, 2014. 2

[27] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, pages 1653–1660, 2014. 2

[28] C. Wei-Chen and M. Fritz. See the difference: Direct pre-image reconstruction and pose estimation by differentiating hog. In *ICCV*, 2015. 2

[29] M. Weinmann, J. Gall, and R. Klein. Material classification based on training data synthesized using a BTF database. In *ECCV*, pages 156–171, 2014. 2

[30] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *CVPR*, 2015. 2

[31] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. *CoRR*, 2014. 2

[32] N. Zhang, J. Donahue, R. B. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, pages 834–849, 2014. 2

[33] M. Zhu, K. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3d object detection and pose estimation for grasping. In *ICRA*, 2014. 1