2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Depth-aware Convolutional Neural Networks for accurate 3D Pose Estimation in RGB-D Images

Lorenzo Porzi[1], Adrian Penate-Sanchez[2], Elisa Ricci[3], Francesc Moreno-Noguer[1]

Abstract— **Most recent approaches to 3D pose estimation from RGB-D images address the problem in a two-stage pipeline. First, they learn a classifier –typically a random forest– to predict the position of each input pixel on the object surface. These estimates are then used to define an energy function that is minimized w.r.t. the object pose. In this paper, we focus on the first stage of the problem and propose a novel classifier based on a depth-aware Convolutional Neural Network. This classifier is able to learn a scale-adaptive regression model that yields very accurate pixel-level predictions, allowing to finally estimate the pose using a simple RANSAC-based scheme, with no need to optimize complex ad hoc energy functions. Our experiments on publicly available datasets show that our approach achieves remarkable improvements over state-of-the-art methods.**

I. Introduction

In recent years, the problem of detecting textureless objects and estimating their 3D pose from a single RGB-D image has received much attention. Existing approaches can be broadly split into two main categories. On the one hand there are methods that rely on matching templates combining image and range data [1], [2]. However, while these are computationally efficient approaches that achieve real time operation, their performance drops under the presence of occlusions. This is addressed by methods which, on the other hand, do not try to find the object as a whole. These methods, instead, first build a classifier that densely predicts the location of each image pixel with respect to an object coordinate system, and then use these predictions to estimate the object's pose in a geometric validation stage [3], [4]. Drawing inspiration from [5], the classifier used so far to regress object pixels into object coordinates is based on random forests. These classifiers, though, typically return very weak confidence maps, making it necessary to put a considerable effort in the geometric phase of the algorithm and having to resort to the minimization of complex energy functions [4].

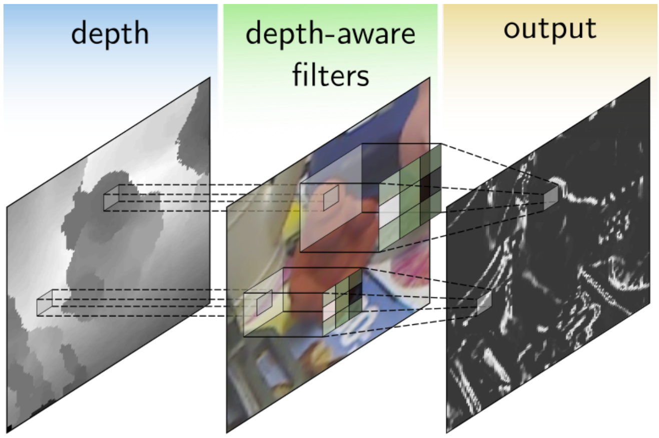

Fig. 1. The intuition behind our novel MultiConv layer: convolutions can be made depth-aware by locally changing the size of the filters depending on the observed depth. Areas of the image corresponding to far away objects are convolved with small filters, while areas corresponding to close objects are convolved with bigger filters.

In this paper, we focus on building a stronger pixel classifier to alleviate the complexity of the subsequent search for geometric consistency. For this purpose, we introduce MultiConv, a novel Convolutional Neural Network (CNN) layer that adapts the size of the convolution filters to the depth values of the input (see Fig. 1). By doing this, we learn a scale-adaptive coordinate regression model, that yields accurate pixel-level object coordinates predictions. The object's 3D pose can then be computed using a simple PROSAC-based strategy. We first demonstrate the effectiveness of our novel MultiConv on semantic segmentation of RGB-D scenes, where our depth-aware CNN performs comparably to specialized state-of-the-art methods. Then, we show how our 3D pose estimation pipeline is able to outperform competing approaches on a publicly available benchmark.

In short, the main contributions of this paper can be summarized as follows:

1) Using a depth-aware CNN for building a coordinate regression model instead of the widely used approach based on random forests [5], [3].
2) Exploiting the predictive power of our CNN to simplify the pipeline for 3D pose estimation from RGB-D images.
3) Introducing a depth-aware CNN architecture. Compared to existing approaches that tackle invariance to generic transformations [6], [7], or scale [8], [9], [10] we use depth information *within* the network as a prior to handle scale.

II. Related Work

Deep learning techniques have began to be applied to robotic tasks improving those tasks in which sufficient

This work is partly funded by the Spanish MINECO project RobInstruct TIN2014-58178-R, by the ERA-Net Chistera project I-DRESS PCIN-2015-147, by the EU project AEROARMS H2020-ICT-2014-1-644271, by the EU project SECOND HANDS H2020-ICT-2014-1-643950 and by the Spanish State Research Agency through the Mara de Maeztu Seal of Excellence to IRI MDM-2016-0656.

[1] Lorenzo Porzi and Francesc Moreno-Noguer are with Institut de Robòtica i Informàtica Industrial (UPC-CSIC), Barcelona, Spain

[2] Adrian Penate-Sanchez is with University College London

[3] Elisa Ricci is with University of Perugia, Italy, and with Fondazione Bruno Kessler, Trento, Italy

978-1-5386-2682-5/17/$31.00 ©2017 European Union

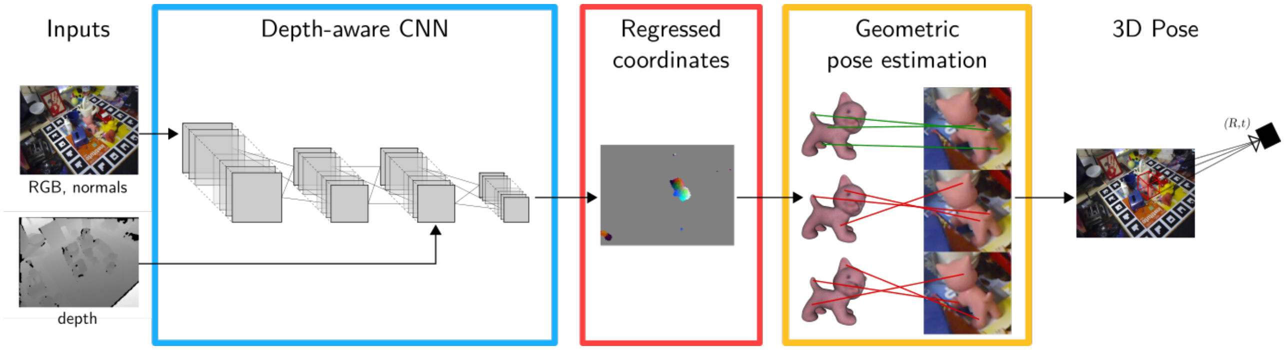

Fig. 2. Algorithm pipeline. The algorithm takes as input an RGB image and its depth map (which is converted to a normal map), and feeds them to a depth-aware CNN. The CNN densely predicts, for every pixel, its corresponding position in the object coordinate framework. Misclassified pixels are rejected and the final pose is estimated using a geometric validation algorithm based on PROSAC. The whole process is executed in less than 200 ms.

amounts of training samples are available. Vital tasks undertaken by robotic systems can benefit from deep learning; we have seen promising improvements to camera relocalization[11], [12], [13], de-convolutional networks are being applied to structural change detection over time in SLAM systems[14], object recognition in point clouds and RGB-D images[15], [16]. This steps forward are showing that robotic vision can greatly benefit from deep learning techniques. We will further detail the state of the art of both 3D pose estimation and deep learning on 3D data.

A. RGB-D approaches to 3D pose estimation

The advent of affordable RGB-D cameras has led to a number of different techniques to detect rigid objects in cluttered environments and estimate their 3D pose. The most straightforward approach is based on *template matching* [17], [2]. Templates pre-acquired from different viewpoints are scanned across the image. At each position, a distance is computed and the best match is retained. However, while this approach is potentially very efficient and can handle textureless objects, the use of a global object model makes it vulnerable to occlusions.

This is addressed by *sparse feature-based methods* [18], which first extract points of interest from the input image, and then match them with the object points using a robust geometric approach [19], [20]. The drawback of these alternatives is that they rely on feature points, extracted either on the RGB [21] or depth domain [22], which makes them only appropriate for objects with a sufficient level of texture or geometric detail.

A different strategy, that has been shown to overcome the limitations of template and feature-based methods, is that used in the so-called *dense approaches*. These methods build classifiers to produce specific per-pixel predictions. The most general approach follows a Hough voting scheme, in which pixels vote for an object pose in a quantized pose space. The region of the pose space with a maximum number of votes is chosen [23], [24]. In [25], latent-class Hough forests are used to let the pixels vote for small templates that cover object parts. More recently, random forests have been used to infer a pixel-level prediction of the observed position in the object's coordinates frame [3]. Such a high level of detail in the prediction is quite challenging and dictates to combine the pixel-level prediction with sophisticated geometric post-processing operations involving the minimization of an energy function. In [4], a CNN that compares real and rendered images is used to learn the aforementioned energy function. One of the main goals of our approach is to avoid these complexities by proposing a better pixel-based classifier, which we build using a novel CNN architecture.

B. CNNs for 3D data

Due to the impressive results achieved in tasks such as object recognition and detection, in the last few years Convolutional Neural Networks have imposed themselves as the main learning paradigm in computer vision, and have recently been used to tackle challenging problems involving 3D data. For instance, Wu *et al.* [26] adopted a CNN-based method to solve next-best-view and depth-based object recognition. Gupta *et al.* [27] used CNNs to learn how to align synthetic 3D models to real instances of the same object in RGB-D scenes, obtaining a significant improvement over the previous work [28], not considering CNNs. Similarly, in the context of feature learning for RGB-D object recognition, Wang *et al.* [29] demonstrated that a CNN-based approach is advantageous over traditional learning based techniques [30].

A recent line of research on CNNs has addressed the problem of devising specific solutions for obtaining invariance to different kinds of transformations. For instance, in [7] a wavelet scattering network, *i.e.* a neural network where the first layer outputs SIFT-like descriptors, is proposed, achieving translational and rotational invariance. Gens *et al.* [31] sought invariance to pose and part deformations and proposed deep symmetry networks. Many works have focused on achieving invariance to scale changes, either by doing a multi-scale pooling [32] of neural activations, or by concatenating the activations obtained from scaled versions of the input before feeding it to the last layers of the network [9]. Differently to these, our method uses depth as a prior to handle scale. Specific efforts to define a common framework for CNN architectures focusing on learning invariant representations has been made in [33] with the introduction of the *Spatial Transformer* layer, which automatically learns a spatial transformation of its input and has some conceptual similarities to the MultiConv layer we propose in this work.

The notion of *Depth-aware Convolutions* was introduced recently in [34]. This work demonstrated the advantages of using depth information during trainning to obtain better results in pixel classification. In contrast to [34] we do not learn the best scale for each convolutional filter, in our case we use the depth information to make the convolutional filters adjust their scale depending on the distance to the object. By doing so we aim to ease the learning uncertainty while at the same time introducing scale invariant properties in the pixel classification that each convolutional filter performs. The use of depth information to achieve invariance to scale changes has been used several times, *e.g.* in conjunction with random forests [35], [3], [5]. In these methods, depth is used to determine the scale at which the binary features of a decision forest are calculated. More recently, similar techniques have also been used in the context of deep learning: in [8], a global depth-dependent scaling is applied to the input of a CNN to solve a segmentation task. In contrast, our approach uses depth *within* the network to locally handle scale at the convolutional filter level.

III. Method

In this work we aim to estimate the pose of an object, of which a 3D model is known, given a single RGB-D image. To do this, we adopt an algorithm in two steps: *coordinates regression* and *geometric pose estimation*. In the first step, for each pixel in the input image we predict its 3D coordinates in the object's frame of reference, using the CNN-based approach described in Section III-A. In the second step, we use the CNN's output in the PROSAC-based [36] procedure described in Section III-B to estimate the object's pose w.r.t. the camera. Fig. 2 shows a visual depiction of our algorithm pipeline.

In this setting it is important to take into account the fact that the object of interest can appear at many different scales, depending on the distance from the camera. To handle this, we introduce MultiConv: a novel CNN layer which performs a *locally multi-scale*, *depth-dependent* convolution operation. Thanks to this layer, our network is able to learn a scale-adaptive coordinates regression model which noticeably improves the accuracy of our approach. More details about MultiConv are presented in Section III-A.2.

A. Depth-aware CNN for coordinates regression

As a first step in our object pose estimation pipeline, we aim to predict, for each pixel in an input image, whether it lies on the object of interest or on the background. Furthermore, if the pixel belongs to the object, we want to predict its 3D coordinates on the object itself. We pose this as a multi-class classification problem, where the pixels of image $\mathbf{I} \in \mathcal{I} = \mathbb{R}^{\mathsf{h} \times \mathsf{w} \times \mathsf{c}}$ with depth $\mathbf{D} \in \mathcal{D} = \mathbb{R}^{\mathsf{h} \times \mathsf{w}}$ are assigned labels $\mathbf{L} \in \mathcal{L} = \{0, 1, \dots, \mathsf{n}\}^{\mathsf{h} \times \mathsf{w}}$. By using the notation $\mathbf{A}[\cdot]$ to indicate indexing into a tensor $\mathbf{A}$, $\mathbf{L}[i, j] = 0$ means that the pixel at coordinates i, j is part of the background, while $\mathbf{L}[i, j] = 1, \dots, \mathsf{n}$ means that the pixel belongs to one of n uniform spatial bins over the object's 3D coordinates.

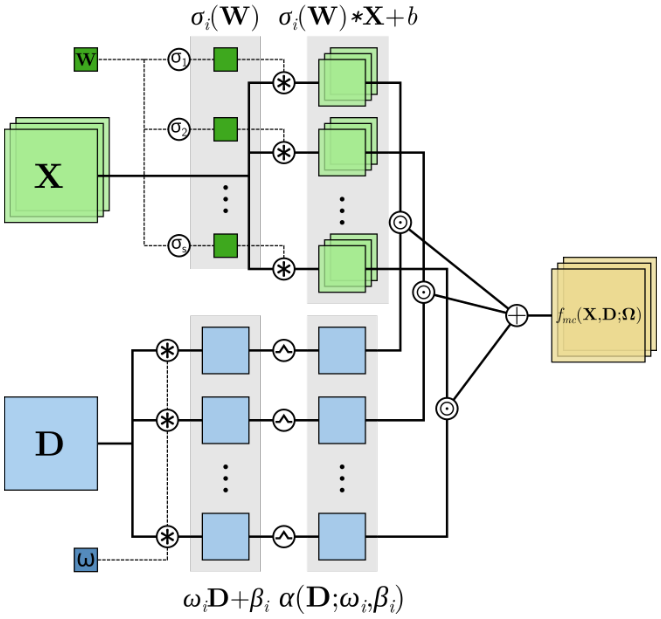
$\sigma_i(\mathbf{W})$
$\sigma_i(\mathbf{W}) * \mathbf{X} + b$
W
σ_1
σ_2
σ_s
X
D
ω
$f_{mc}(\mathbf{X},\mathbf{D};\mathbf{\Omega})$
$\omega_i\mathbf{D}+\beta_i$
$\alpha(\mathbf{D};\omega_i,\beta_i)$

Fig. 3. Schematic representation of the MultiConv layer.

We model the relation between the image and the pixels' labels using a fully-convolutional neural network. We write the network as a function $f_{\text{CNN}} : \mathcal{I} \times \mathcal{D} \rightarrow \mathbb{R}^{\mathsf{h} \times \mathsf{w} \times (\mathsf{n}+1)}$ with parameters $\mathbf{\Theta}$, such that

$$\mathbf{Y}[i, j, k] = \mathrm{P}(\mathbf{L}[i, j] = k \mid \mathbf{I}, \mathbf{D}, \mathbf{\Theta}) \quad \forall i, j, k, \qquad (1)$$

where $\mathbf{Y} = f_{\text{CNN}}(\mathbf{I}, \mathbf{D}; \mathbf{\Theta})$. As it is common with CNNs, we learn $\mathbf{\Theta}$ by minimizing a regularized log-loss function over a training set of image-depth-label triplets $(\mathbf{I}, \mathbf{D}, \mathbf{L}) \in \mathcal{T} \subset \mathcal{I} \times \mathcal{D} \times \mathcal{L}$.

As an input to our network, we use image tensors with $\mathsf{c} = 6$ channels, specifically: red, green and blue color intensities and x, y and z components of the surface normal vectors. The normals are calculated analytically from a bicubic fit on the point cloud generated by the depth data.[1]

1) Network architecture: Contrary to common CNN architectures, we are interested in obtaining a *dense* labeling over the pixels of the input image, instead of a single, global label. To achieve this we adopt a fully convolutional approach, where each layer of the network operates convolutionally over its input. Table I shows the detailed structure of our network. Note that the final softmax is also applied independently on each spatial location, *i.e.* it is a function $f_{\text{sm}}(\cdot)$ such that:

$$\mathbf{Y} = f_{\text{sm}}(\mathbf{X}) \Rightarrow \mathbf{Y}[i, j, k] = \frac{e^{\mathbf{X}[i,j,k]}}{\sum_k e^{\mathbf{X}[i,j,k]}} \ \forall i, j, k \ , \qquad (2)$$

where $\mathbf{X}$ is the output tensor of the previous layer.

Traditional CNN architectures often adopt an aggressive internal down-sampling of the data, obtained by striding the

[1] In practice we use MATLAB's `surfnorm` function.

#	layer	size	stride	pre-training
1.1	conv	$3 \times 3 \times 16$	1	✓
1.2	conv	$3 \times 3 \times 16$	1	×
1.3	max	2×2	2	✓
2.1	conv	$3 \times 3 \times 32$	1	✓
2.2	conv	$3 \times 3 \times 32$	1	×
2.3	max	2×2	2	✓
3.1	mc	$3 \times 3 \times 64$	1	✓
3.2	conv	$3 \times 3 \times 64$	1	×
3.3	max	2×2	1	✓
4.1	conv	$1 \times 1 \times 64$	1	×
4.2	drop	–	–	✓
4.3	conv	$1 \times 1 \times (\mathsf{n}+1)$	1	✓
4.5	sm	–	–	✓

TABLE I

THE ARCHITECTURE OF OUR CNN. WE USE THE FOLLOWING CONVENTION FOR THE LAYERS' NAMES: CONVOLUTION (CONV), MAX-POOLING (MAX), MULTICONV (MC), DROP-OUT (DROP), SOFTMAX (SM). A CHECK MARK IN THE LAST COLUMN INDICATES THAT THE LAYER IS USED IN THE PRE-TRAINING PHASE.

convolutions and pooling masks with steps greater than 1. This helps to keep in check the memory and computational requirements of the net, while allowing for *wider* layers (*i.e.* layers with many filters). In our case, the requirement for a dense output clashes with this commonly adopted trick. As a compromise, we set the stride of the two max-pooling layers 1.3 and 2.3 to 2, resulting in a final down-sampling factor of 4. Furthermore, we pad with zeros the input of each other convolution and pooling layer as appropriate to maintain the spatial size of their outputs equal to that of the inputs. The overall effect is that each element of the network's output corresponds to a 4×4 pixels area of the input image.

Our architecture is inspired by the "very deep" networks of Simonyan and Zisserman [37], [38]. In this kind of nets, compared to traditional ones, bigger convolutions are replaced with blocks of cascaded convolutions of smaller sizes (usually 3×3). This increases the overall non-linearity while decreasing the number of parameters, at the cost of a resulting network that is more difficult to train. As in [37], we adopt a two-steps training procedure: first we train a *shallow* version of the network containing a subset of the layers (see pre-training column in Table I), then we add the remaining layers and complete the training. Both training phases are carried out using mini-batch stochastic gradient descent with momentum.

2) MultiConv layer: The MultiConv layer performs a *locally multi-scale*, *depth-dependent* convolution operation, where the relation between depth and scale is learned together with the convolution parameters. For each spatial location on the input, MultiConv performs the same convolution at s different scales, then linearly combines the results using a set of weights that are functions of the depth. Figure 3 shows a schematic representation of this approach. More specifically, we express the output of MultiConv as a function $f_{\text{mc}}(\mathbf{X}, \mathbf{D}; \mathbf{\Omega})$, where $\mathbf{X}$ and $\mathbf{D}$ are, respectively, the input tensor and the depth. $\mathbf{\Omega}$ is a tuple of parameters to be learned $\mathbf{\Omega} = (\mathbf{W}, b, \omega_1, \ldots, \omega_{\mathsf{s}}, \beta_1, \ldots, \beta_{\mathsf{s}})$, with $\mathbf{W}$ and b being the convolution weights and bias, respectively, and $\{\omega_i, \beta_i\}$, $i = 1, \ldots, \mathsf{s}$ the weights that linearly combine the depth entries. The function $f_{\text{mc}}(\cdot)$ can then be formally written as:

$$f_{\text{mc}}(\mathbf{X}, \mathbf{D}; \mathbf{\Omega}) = \sum_{i=1}^{\mathsf{s}} \alpha(\mathbf{D}; \omega_i, \beta_i) \odot (\sigma_i(\mathbf{W}) * \mathbf{X} + b), \quad (3)$$

where $*$ denotes convolution and $\odot$ denotes element-wise multiplication[2]. $\sigma_i(\cdot)$, $i = 1, \ldots, \mathsf{s}$ are a set of filter scaling functions defined as:

$$\sigma_i(\mathbf{W}) = (\mathbf{W} \uparrow 2^{i-1}) * \mathbf{G}_i, \quad (4)$$

where $(\cdot \uparrow N)$ denotes the 2D stretch operator, which intersperses the elements of its left operand with $N-1$ zeros along each spatial dimension, while $\mathbf{G}_i$ is a Gaussian filter with variance 2^{i-1}.

The function $\alpha(\cdot)$ in (3) is a depth-dependent weighting function defined as:

$$\alpha(\mathbf{D}; \omega_i, \beta_i) = \operatorname{tri}(\omega_i \mathbf{D} + \beta_i), \quad (5)$$

where $\operatorname{tri}(\cdot)$ is the "triangle" function:

$$\operatorname{tri}(t) = \max\{0, 1 - |t|\}. \quad (6)$$

The intuition behind (5) is that we expect each scale to be most appropriate for a specific depth, while decreasing in importance as the depth changes. Keeping this in mind, it is easy to see that, by learning β_i and ω_i, MultiConv chooses a preferred depth for each scale i, corresponding to the maximum of $\alpha(d; \beta_i, \omega_i)$ at $d = -\frac{\beta_i}{\omega_i}$. For other values of d, the convolution at scale i gets assigned a weight $\alpha(d; \omega_i, \beta_i) > 0$ as long as $\frac{-1-\beta_i}{\omega_i} < d < \frac{1-\beta_i}{\omega_i}$.

As a final note, we point out that the derivatives of MultiConv are immediate to calculate after noting that $f_{\text{mc}}(\cdot)$ is separately linear in $\mathbf{X}$, $\mathbf{W}$ and b, and the derivative of $\operatorname{tri}(\cdot)$ is:

$$\frac{d \operatorname{tri}(t)}{dt} = \begin{cases} 1 & -1 < t < 0 \\ -1 & 0 < t < 1 \\ 0 & t < -1 \vee t > 1 \end{cases}. \quad (7)$$

B. Geometric pose estimation

The *geometric pose estimation* step of our pipeline estimates the object's pose by minimizing a geometric error function defined in terms of 3D-to-3D point correspondences between the camera's and the object's frame of reference.

Let us assume a pin-hole projective camera model with focal lengths (f_x, f_y) and central point (c_x, c_y). We can then reconstruct the 3D coordinates, in camera's reference frame, of a pixel at position i, j on the image $\mathbf{I}$ with depth $\mathbf{D}$ as:

$$\mathbf{p}_{i,j}^{\mathcal{C}} = \frac{\mathbf{D}[i,j]}{\sqrt{1 + \hat{x}^2 + \hat{y}^2}} \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix}, \quad (8)$$

where

$$\hat{x} = \frac{j - c_x}{f_x}, \quad \hat{y} = \frac{i - c_y}{f_y}.$$

[2] Note that we are assuming that $\sigma_i(\mathbf{W}) * \mathbf{X}$ has the same spatial size as $\mathbf{D}$. In practice, we can always match $\mathbf{D}$ to the convolution's output by properly scaling it and cropping its borders to account for padding.

Sequence	CNN-basic	CNN-MC
ape	31.2	26.7
benchvise	45.5	37.3
bowl	74.2	58.2
cam	45.4	35.6
can	48.2	36.9
cat	39.4	32.8
cup	65.0	53.4
driller	48.1	38.6
duck	36.6	29.0
eggbox	34.5	25.0
glue	37.2	29.9
holepuncher	44.2	33.6
iron	46.7	34.5
lamp	54.4	45.7
phone	46.7	36.1

TABLE II

Pixel classification error (%) on the Hinterstoisser dataset

From the CNN's output $\mathbf{Y} = f_{\text{CNN}}(\mathbf{I}, \mathbf{D}; \mathbf{W})$, we obtain a set of pixels $\mathcal{P}$ that are predicted to have a high probability of being on the object by imposing a threshold $\tau > 0.5$:

$$\mathcal{P} = \{(i, j) \mid \textstyle\sum_{k>0} \mathbf{Y}[i, j, k] > \tau\}. \tag{9}$$

Each of these pixels is assigned a label

$$l_{i,j} = \arg\max_{k} \mathbf{Y}[i, j, k], \tag{10}$$

which corresponds to a certain spatial bin on the object's 3D coordinates. We denote the centroid of the k-th bin as $\mathbf{c}_k^{\mathcal{O}}$, expressed in the object's frame of reference. Thus, each point in $\mathcal{P}$ is predicted to have 3D coordinates in the object's frame of reference given by $\mathbf{p}_{i,j}^{\mathcal{O}} = \mathbf{c}_{l_{i,j}}^{\mathcal{O}}, \forall (i, j) \in \mathcal{P}$.

We estimate the object's pose, expressed as a rotation-translation pair $(\mathbf{R}_{\mathcal{O}}^{\mathcal{C}}, \mathbf{t}_{\mathcal{O}}^{\mathcal{C}})$, by solving the following optimization problem:

$$\arg\min_{\mathbf{R}_{\mathcal{O}}^{\mathcal{C}}, \mathbf{t}_{\mathcal{O}}^{\mathcal{C}}} \sum_{(i,j)\in\mathcal{P}} \|\mathbf{R}_{\mathcal{O}}^{\mathcal{C}} \mathbf{p}_{i,j}^{\mathcal{O}} + \mathbf{t}_{\mathcal{O}}^{\mathcal{C}} - \mathbf{p}_{i,j}^{\mathcal{C}}\|^2. \tag{11}$$

Even though the pixel classification error of our depth-aware CNN is very low, we still need to handle a certain amount of outliers (below 35% in most of the experiments we report in the following section). For such a misclassification rate, Eq. (11) can be safely solved using a simple outlier rejection approach like the PROSAC algorithm [36]. This is a RANSAC variant that generates hypotheses and tests them on subsets of points sorted by their class probability. In our case, the class probability is given by the CNN prediction. This geometric validation stage turned out to converge very fast in our case, adding almost no time penalty to the overall process.

IV. Experiments

To evaluate our approach we performed a series of experiments to evaluate the performance of our method on the 3D pose estimation dataset from Hinterstoisser *et al.* [2]. In the following we denote the proposed CNN architecture with the MultiConv layer as **CNN-MC**, in which we use s $= 3$ scales in the MultiConv layer. To demonstrate the validity of our proposal we also consider an additional architecture, denoted as **CNN-basic**, corresponding to a CNN as described in Table I but with the MultiConv layer replaced by a convolutional layer of the same size.

	Brachman *et al.* [3]		CNN-basic		CNN-MC	
Sequence	tran.	rot.	tran.	rot.	tran.	rot.
ape	7.8	10.1	3.8	5.5	3.7	5.9
benchvise	9.4	7.0	7.3	4.1	6.6	3.3
cam	10.9	10.9	6.9	5.9	6.6	5.1
can	8.6	6.3	7.9	5.3	7.1	5.1
cat	7.5	5.3	3.9	4.1	3.5	4.0
driller	12.6	8.9	7.3	2.9	5.9	2.7
duck	7.0	7.9	3.9	6.5	3.4	6.4
eggbox	7.5	5.2	3.0	4.1	2.6	4.2
glue	9.4	11.8	6.2	5.3	6.4	5.3
holepuncher	6.2	5.6	5.0	5.0	4.4	3.6
iron	–	–	7.4	4.6	7.5	3.9
lamp	15.8	12.2	8.1	4.2	7.1	3.2
phone	–	–	5.8	4.5	5.1	4.0
Total:	9.6	9.1	5.6	4.7	5.1	4.3

TABLE III

Detailed pose estimation results on the Hinterstoisser dataset: median translation and rotation error divided by sequence. Note: for two of the objects (bowl and cup) the dataset does not provide a proper 3D model, and neither our approach, nor Brachman's can be applied.

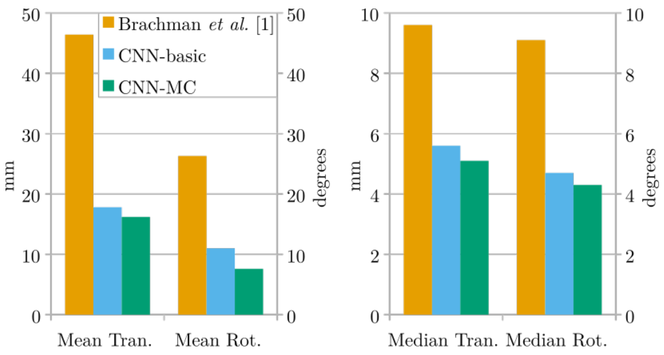

Fig. 4. Pose estimation results on the Hinterstoisser dataset: mean and median translation and rotation error on the whole dataset.

The 3D object pose estimation dataset from Hinterstoisser *et al.* [2] contains colored 3D models associated to 15 textureless objects and 15 video sequences, each containing about 1,000 RGB-D frames, depicting the objects on a cluttered desk. In each sequence ground truth pose information is given for one of the objects. This dataset is suitable to assess the validity of the proposed method as the test images cover the upper view hemisphere at different scales. In our experiments, we use 80% of the RGB-D frames (chosen at random) to train our CNN and the remaining 20% to test the pose estimation pipeline. In performing our experiments we came about some issues in the ground truth annotations given in the Hinterstoisser dataset. In particular, for a small number of the objects, the ground truth pose in many of the frames was noticeably inconsistent with the depth data.

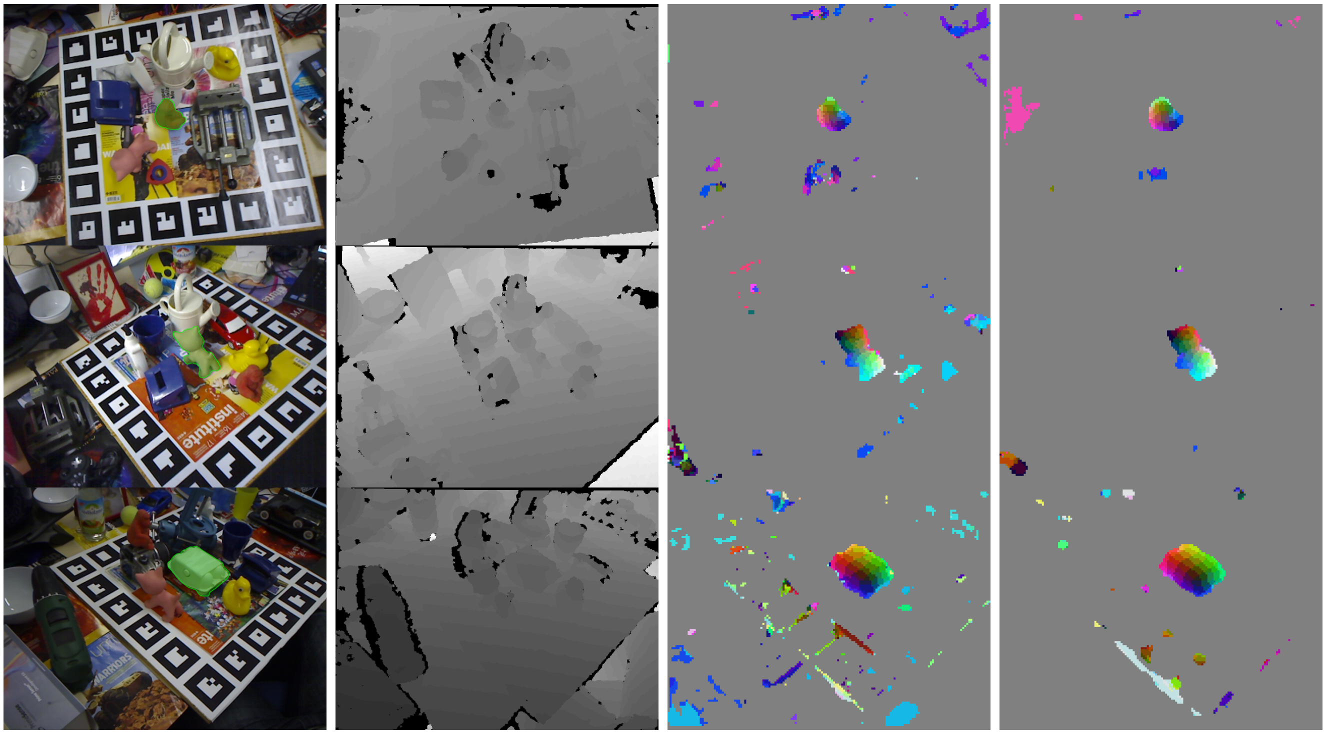

Fig. 5. CNN coordinates regression output. First column: input image. The detected object is overlaid using the estimated 3D pose. Second column: input depth map. Third column: Object 3D coordinates mapped to the RGB color space. Gray pixels are those classified as background. Results obtained for the CNN-basic architecture. Fourth column: The same for CNN-MC. Observe how handling scale with CNN-MC noticeably decreases the number of outliers compared to CNN-basic.

We hypothesize that this was caused by some sort of error when annotating the ground truth of those objects. Since our method strongly relies on the depth when estimating the object's pose, we tried to fix the inconsistencies by performing an ICP alignment on the ground truth of the most problematic frames. In the following, all results are obtained by running our methods and the baselines on the "fixed" data. This fixed data will be made publicly available to facilitate future research.

We first performed some preliminary experiments on pixel classification to demonstrate the advantages of the proposed CNN-MC over CNN-basic. Table II shows the results of our comparison. It is clear that embedding a scale-adaptive scheme into a CNN architecture significantly reduces the error for all the experiments on different objects.

We then evaluated the accuracy of the proposed approach in pose estimation. Following previous works [2], [3], we consider one object per image and we assume to know which object is present in the scene. The proposed approach is compared with the state of the art method in [3]. For [3] we used the publicly available binaries[3].

Most previous works on the Hinterstoisser dataset showed their results in terms of the percentage of frames in which a measure of the proximity between the ground truth object model and the estimated one is lower than a certain threshold. Conversely, we report the mean and median translation and rotation errors. We believe that these measures give a more direct, and thus more significant evaluation of the pose estimation accuracy.

Detailed results on the single objects are shown in Table III. Note that we omitted some entries from the Brachmann *et al.* columns, as we were not able to reproduce their results on the phone and iron objects. We also omitted the bowl and cup objects, as the dataset did not provide their 3D mesh model. Figure 4 summarizes the mean and median error results we obtained on the whole dataset. It is clear that our pose estimation approach outperforms the method in [3]. Moreover, similarly to what was observed in the semantic segmentation experiments, learning the parameters ω_i and β_i in the MultiConv layer is beneficial. We would like to mention that the recent work [4], reports a remarkable improvement w.r.t. [3]. Unfortunately, the code for this new approach is not still available and we were not able to include it in our comparison. That being said, we believe that our approach and [4] would be complementary, as the latter is focused on improving the geometric validation phase of the problem, while we focus on robustifying the initial pixel classification.

Finally, Fig. 5 shows some qualitative results associated to our experiments. Also in this case it is possible to observe that more accurate pixel-level predictions can be obtained with our scale-adaptive CNN-MC over CNN-basic.

[3] http://cvlab-dresden.de/research/scene-understanding/pose-estimation/#ECCV14

V. Conclusion

We have presented a novel depth-aware CNN for pixel level classification in RGB-D images. The classifier has been shown to be adequate when used as a regressor for a 3D pose estimation problem, by predicting, for each pixel of the input image, its 3D coordinates in the object coordinate frame. Since these predictions are in general very accurate and contain small amounts of misclassifications, they allow for a simple outlier rejection scheme to finally estimate the object pose. Results over existing baselines show to consistently improve state-of-the-art approaches that use less confident pixel predictors, but more elaborate outlier rejection algorithms than we do. Future steps involve experimenting with solutions to impose geometric consistency directly in the CNN output, *e.g.* by integrating a CRF-based approach into the network [39].

References

[1] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2011.

[2] 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 *Asian Conf. on Computer Vision (ACCV)*, 2013.

[3] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European Conf. on Computer Vision (ECCV)*, 2014.

[4] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6d pose estimation in rgb-d images," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2015.

[5] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[6] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Neural Information Processing Systems (NIPS)*, 2015.

[7] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 35, no. 8, pp. 1872–1886, 2013.

[8] H. Schulz, N. Höft, and S. Behnke, "Depth and height aware semantic rgb-d perception with convolutional neural networks," in *European Symp. on Artificial Neural Networks (ESANN)*, 2015.

[9] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," in *Int. Conf. on Learning Representations (ICLR)*, 2013.

[10] N. Höft, H. Schulz, and S. Behnke, "Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks," in *German Conf. on Artificial Intelligence (KI)*, 2014.

[11] N. Suenderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford, "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free," in *Robotics: Science and Systems Conference (RSS)*, 2015.

[12] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[13] A. Rubio, M. Villamizar, L. Ferraz, A. Penate-Sanchez, A. Ramisa, E. Simo-Serra, A. Sanfeliu, and F. Moreno-Noguer, "Efficient monocular pose estimation for complex 3d models," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[14] P. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi, "Street-view change detection with deconvolutional networks," in *Robotics: Science and Systems Conference (RSS)*, 2016.

[15] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[16] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[17] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 34, no. 5, pp. 876–888, 2012.

[18] M. Martinez, A. Collet, and S. S. Srinivasa, "Moped: A scalable and low latency object recognition and pose estimation system," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[20] F. Moreno-Noguer, V. Lepetit, and P. Fua, "Accurate non-iterative o (n) solution to the pnp problem," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2007.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[22] S. Holzer, J. Shotton, and P. Kohli, "Learning to efficiently detect repeatable interest points in depth data," in *European Conf. on Computer Vision (ECCV)*, 2012.

[23] M. Sun, G. Bradski, B.-X. Xu, and S. Savarese, "Depth-encoded hough voting for joint object detection and shape recovery," in *European Conf. on Computer Vision (ECCV)*, 2010.

[24] O. J. Woodford, M.-T. Pham, A. Maki, F. Perbet, and B. Stenger, "Demisting the hough transform for 3d shape recognition and registration," *Int. J. of Computer Vision (IJCV)*, vol. 106, no. 3, pp. 332–341, 2014.

[25] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *European Conf. on Computer Vision (ECCV)*, 2014.

[26] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, "3d shapenets for 2.5 d object recognition and next-best-view prediction," *arXiv preprint arXiv:1406.5670*, 2014.

[27] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, "Aligning 3d models to rgb-d images of cluttered scenes," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[28] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conf. on Computer Vision (ECCV)*, 2014.

[29] A. Wang, J. Cai, J. Lu, and T.-J. Cham, "Mmss: Multi-modal sharable and specific feature learning for rgb-d object recognition," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2015.

[30] A. Wang, J. Lu, G. Wang, J. Cai, and T.-J. Cham, "Multi-modal unsupervised feature learning for rgb-d scene labeling," in *European Conf. on Computer Vision (ECCV)*, 2014.

[31] R. Gens and P. M. Domingos, "Deep symmetry networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conf. on Computer Vision (ECCV)*, 2014.

[33] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[34] L. Porzi, S. Rota-Bulo, A. Penate-Sanchez, E. Ricci, and F. Moreno-Noguer, "Learning depth-aware deep representations for robotic perception," *IEEE Robot. and Autom. Let. (RA-L)*, vol. 2, no. 2, pp. 468–475, 2017.

[35] A. C. Muller and S. Behnke, "Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[36] O. Chum and J. Matas, "Matching with prosac-progressive sample consensus," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. on Learning Representations (ICLR)*, 2015.

[38] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman, "Deep convolutional neural networks for efficient pose estimation in gesture videos," in *Asian Conf. on Computer Vision (ACCV)*, 2015.

[39] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional random fields as recurrent neural networks," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2015.