On The Fly Generated Data for Industrial Part Orientation Estimation with Deep Neural Networks

Julien Langlois, Harold Mouchère, Nicolas Normand, Christian Viard-Gaudin

On The Fly Generated Data for Industrial Part Orientation Estimation with Deep Neural Networks

J. Langlois[a,b], H. Mouchère[b], N. Normand[b] and C. Viard-Gaudin[b]

[a]Multitude-Technologies, a company of Wedo
[b]LS2N UMR 6004, University of Nantes

ABSTRACT

In this paper a method is proposed for estimating the orientation of industrial parts. Classical 2D images of the part are used to train a deep neural network and infer the part pose by computing its quaternion representation. Another innovative point of this work is in the use of synthetic data which are generated on the fly during the network training from a textured CAD model placed in a virtual scene. This way overcomes the difficulties to obtain pose ground truth images from real images. At the same time, using the CAD model with several lighting conditions and material reflectances allows to anticipate challenging industrial situations. As a first step of the method, the part is separated from the background using a semantic segmentation network. Then a depth image of the part is produced employing an encoder-decoder network with skip connections. Finally, the input image is associated with its depth map to estimate the part orientation with a fully connected network using a geodesic loss function. This method can estimate the part pose from a rendered image with a 5° accuracy while inferring from real images gives visually convincing results suitable for any pose refinement processes.

Keywords: Pose estimation, deep neural networks, convolutional neural networks, depth estimation, industrial parts, rendered data

1. INTRODUCTION

Object pose estimation is a crucial functionality for various problems such as virtual reality or bin picking with robotic grasping. It remains a challenging task when the parts are poorly textured and have complex shapes. While several methods can rely on simple pattern detection for a pose estimation problem, the performances falls when the industrial environment is not light controlled or known a priori. Besides, any black and glossy plastic material can be difficult to handle because of the image low contrast and sometimes the light saliency on the part surface. Moreover, this task remains challenging when only classical RGB images are available. During the last years, convolutional neural networks (CNNs) appeared relevant for the pose estimation problem. Their strong features learning abilities are essential to build a mapping from an object RGB image to its orientation in $SO(3)$.

Figure 1. Full pose estimation pipeline from left to right : the image of an object on a conveyor is first segmented then the local depth is inferred before estimating the orientation and computing the final pose in $SE(3)$.

In this paper, a neural-based orientation estimation technique is presented using only one RGB image of a scene containing an object of interest. Lots of samples are required to learn a RGB to $SO(3)$ mapping and build a real image dataset can be time consuming. Moreover, an object inside a bin can be placed against other parts so that its orientation, can be any element of $SO(3)$ which is difficult to obtain for a ground truth. To tackle this issue, the training dataset is composed with only rendered images. Using a data generation process enhances the dataset versatility by rendering random poses with various light conditions. When using an opengl shader in a screenless context, the image generation may be done on-the-fly during the training phase thus, no storage is required. The proposed pipeline for the pose estimation is divided in three distinct neural networks respectively segmentation, depth and orientation networks. The parts are assumed to be at a fixed distance from the camera lying on the same plane (eg. a conveyor) without any occlusions. The translations on both x and y can be easily obtained using each object pixel coordinates.

This paper is divided as follows. First related works dealing with neural-based pose estimation are presented (Sec. 2). The image generation technique is then introduced (Sec. 3) followed by the segmentation technique (Sec. 4). The depth (Sec. 5) and orientation (Sec. 6) estimations are detailed before the experiments (Sec. 7) and conclusions (Sec. 8).

2. RELATED WORKS

Deep learning approaches for pose estimation have been widely studied in the literature the past years for they provide useful pose dependent features from object images. One of the first work dealing with object pose estimation as a classification,[1] employs a simple k-NN algorithm working on features extracted from a convolutional neural network (CNN). The learning phase is done using synthetic object RGB-D images, obtained with an even camera placement on an encompassing sphere. The network is fed with image triplets composed of similar and dissimilar poses, to ensure that two close poses have a small distance in the feature space. The feature-based pose classification can also be done using a support-vector machine (SVM).[2] However, the pose estimation acts as a classifier based on a subset discretization of the view space. To get a more detailed pose, a probabilist approach[3] employs a regression-classification random forest to define an energy function for pose estimation and an optimization scheme to minimize the error. Later, the energy function is replaced with a CNN as a probabilistic model to learn a comparison between the input image and a rendered object.[4] The method can also be improved using reinforcement learning.[5] The pose estimation problem can be treated as a regression problem in an end-to-end manner. A Siamese architecture among specific loss function terms, can enforce close poses to have similar features as well as handling occlusions.[6] Nonetheless, these works take profit of the image depth channel as the mapping from RGB to the pose space is complex. Yet, based on the GoogLeNet network,[7] PoseNet[8] computes a camera pose from RGB images suitable for outdoor scene images. To deal with objects, the single shot detector (SSD) framework[9] is adapted to compute a set of object orientations. The object pose among its translation is later classified.[10] However, estimating the pose with an end-to-end method is not always the best solution. Several networks can also be used to generate region of interests then, infer the pose of multiple objects in the image.[11] The technique uses the camera intrinsic parameters to retro-project the object coordinates into the 3D scene. The iPose work[12] proposes to infer the object pixel-wise coordinates from and encoder-decoder network then, compute the pose with a RANSAC algorithm. Rather then inferring the coordinates, the network can provide the image optical flow.[13] In DeepIM,[14] the features of the optical flow network are used to compute the object pose with dense layers reaching the state of the art in the LINEMOD dataset. The training is done with real images from the dataset among rendered views with textures and light distributions extracted from the dataset. Alternatively, the pose can be computed with a PnP algorithm from the object 3D bounding box obtained with a VGG[15] network.[16] These techniques can weather rely on the datasets images or complete the data with synthetic images using the same light distribution. Therefore, they are suitable for dataset challenges but might have difficulties to handle versatile industrial environment not know *a priori*.

3. IMAGE GENERATION

The CAD model of the object is a triangular mesh composed with a set of surfaces $\{s\}_i$ and associated normals $\{\vec{n}\}_i$. Given an image size $H \times W$, the object pose view $I \in [0,1]^{H \times W}$ is obtained with the model surfaces projections on the image plane, using a combination of a translation $\vec{t} \in \mathbb{R}^3$, rotation $R \in SO(3)$ and a camera intrinsic parameters matrix within an openglGL shader. The depth image is collected from the OpenGL normalization of each surface z-coordinate. However, this normalization is first mapping $[z_n, z_n]$ to $[-1,1]$ with a non linear behavior ($\propto \frac{1}{z}$) than transforming it linearly to $[0,1]$. Thus, a depth range set regarding to the object maximum radius r_o and its z-translation t_z as $[t_z - r_o, t_z + r_o]$, is enough to get a detailed depth map $D \in [0,1]^{H \times W}$. Let (f_X, f_Y) be the camera focal lengths in pixels and (c_X, c_Y) the image center, the full projection $(x, y, z) \mapsto (p_x, p_y, \bar{z})$ can be expressed as a tensor product using homogeneous coordinates:

$$\begin{pmatrix} p_x \\ p_y \\ \bar{z} \\ 1 \end{pmatrix} = \begin{pmatrix} 2f_X & 0 & c_x & 0 \\ 0 & -2f_Y & c_y & 0 \\ 0 & 0 & -\frac{t_z}{r_o} & \frac{t_z^2 - r_o^2}{r_o} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} R & \vec{t} \\ \vec{0}^T & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1}$$

Each pixel is colored according to a Phong's shading technique controlled by 3 components respectively ambient L_a, diffuse L_d and specular L_s, a material glowing factor and a light position (Fig. 2). The image background is set to be an alpha channel so that it directly represents the binary segmentation mask and can ease the insertion of future backgrounds. For each generated images, a Gaussian blur is added to smooth the part edges among random noise and gamma variations.

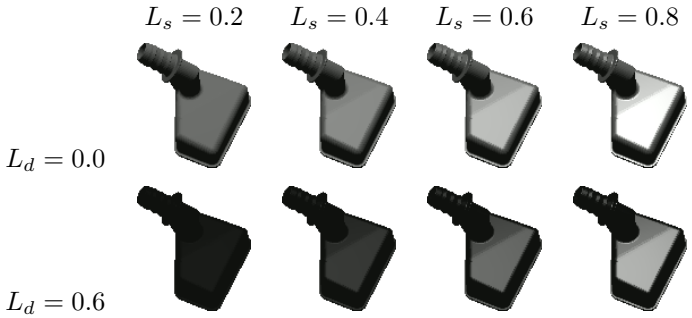

Figure 2. Effects of the light diffuse L_d and specular L_s components variation.

4. PARTS SEGMENTATION

The segmentation process is done using an encoder-decoder network with skip connections. All the layers are using ReLu activation except the last layer which uses a sigmoid activation. The training is done with a simple MSE loss. Segmenting the full-sized image needs a heavy network with a long inference phase which is not suitable for a fast industrial process. On the other hand, using a low resolution segmentation leads to rough results often smoothed using CRFs. In this work, the segmentation network is fed with small patches $P_I \in [0, 1]^{128\times128}$ taken close from the object location on the image. This means that the network is learning to segment an object which is entirely or partially seen in an image. As the networks needs to eliminate the background, some patches without object are also taken into account. The patches background are extracted from a conveyor textures set with random transformations (rotation, flip, noise...). The inference phase is done with a sliding window method where all inferred patches are summed then normalized. This gives a segmentation heat map of the scene that can be thresholded to get the objects binary mask. However, this method can not handle occlusions as it is assumed that all the parts are sequentially coming on a conveyor (not challenging if a proper supplier is used). Various light conditions are applied to the object so that the network can absorb any light saliences (Fig. 3). The instance segmentation is done with a simple contour detection applied on the objects binary mask. Each object is then patched according to its bounding box before being resized to 128×128 using a bilinear interpolation. The patch background is set to -1 as the future depth network only uses the foreground information.

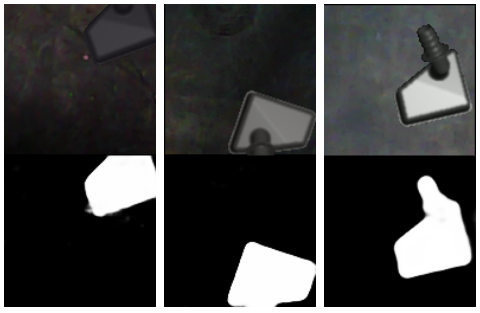

Figure 3. Partial object segmentation inference on virtual image with different light parameters from flat appearance (left) to shiny material (right).

5. DEPTH PREDICTION

The normalized depth prediction is made with another encoder-decoder network with skip connection based on the one use for the segmentation. However, 2 fully connected layers (FC) are now inserted between the encoding and decoding phase enhancing the control of the latent space. The last FC layer output is then reshaped to a matrix to be fed into the first decoder deconvolution (transposed convolution) (Fig. 4). A dropout layer is also placed close to the input to anticipate any previous segmentation artifacts that may disturb the network during the inference. When trying to learn a depth patch P_D from an image patch P_I using a sigmoid activation and a Huber loss function (or MSE), the inference appears to be noisy and the transposed convolutions sometimes give squared artifacts. To tackle this issue, rather than predicting a single channel, the network outputs two channels respectively P_D and $(1 - P_D)$ (so that the channel-wise sum is 1 for every pixel) using a softmax non-linearity. This gives smoother results as the softmax function tends to eliminate the output noises. The encoder-decoder shows a great light saliences robustness even with high contrasts. The loss $\mathcal{L}_D$ is a cross-entropy function working a both output channels pixels (u, v) between a prediction P_{D_i} and its ground truth $\hat{P_{D_i}}$:

$$\mathcal{L}_D = \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{u,v} \left(\hat{P_{D\,i}} \log \left(P_{D\,i} + \epsilon \right) \right) \right) \tag{2}$$

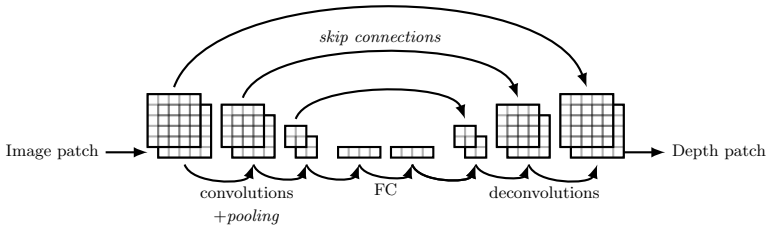

Figure 4. Depth estimation encoder-decoder network architecture using skip connections and dense layers within the latent space.

6. ORIENTATION ESTIMATION

The part orientation is given by a convolutional neural network using 3 convolutional layers and 3 FC layers. The network is fed with the previous depth patch P_D that can be stacked with the image patch P_I. However, if the encoder-decoder architecture infers correctly real images from a synthetic data based training, this is not necessary the case with a CNN. Using the image patch in the orientation network adds another virtual-real data bias in the pipeline and thus, it decreases the orientation estimation performances. The object orientation is expressed as a unitary quaternion $q \in [-1, 1]^4$ where $||q|| = 1$. The quaternion is obtained with a tanh function before getting manually normalized. As q and $-q$ represents the same orientation (obvious when using Rodrigues rotation formula), a L_2 loss is not a valid metric if the network is prediction the opposite of the ground truth. The $SO(3)$ metric between two rotation matrices R_1 and R_2, $||I - R_1^T R_2||$, can be transformed to measure the difference between two quaternions q_1 and q_2 as $2\sqrt{2\left(1 - |q_1.q_2|^2\right)}$ (where . is the inner product). The network loss is then defined as $\mathcal{L}_O$:

$$\mathcal{L}_O = \sqrt{1 - (q_1.q_2)^2} \tag{3}$$

Two image patches of an object with different orientations and plane translations can however be similar. The orientation network is then also fed with the normalized pixel coordinates of the patch $P_C \in [-1, 1]^{128 \times 128}$ in the original image. This allows the network to know where the object is in the camera plane and distinguish the poses despite the images equivalences. Moreover, as the z-translation is known *a priori*, the coordinates retro-projection gives a good approximation of the object x and y-translations in the scene. A refinement technique such as ICP, can be use to align the CAD model within the image.

7. EXPERIMENTS

7.1 Protocol

As the part is supposed to stay on a conveyor, it can only shows balanced poses to the camera. However, in this work, no assumptions are made on the available part orientations. The rendered images are obtained using homogeneous poses on the object encompassing sphere given the two first rotation angles (ϕ, θ). The third angle ψ (camera plane rotation) is randomly taken from the $[-\pi, \pi]$ interval. The x and y-translations are also randomly taken so that the part covers all the camera plane. This means that the networks are able to handle every part orientations without *prior* studies on the part geometry and inertia. The networks are trained for 1K epochs using *adam* optimizer for both segmentation and depth networks and Nesterov momentum for the orientation network with a 10^{-4} learning rate. The weights giving the best results on a rendered validation dataset are kept.

To get a pose estimation error based on real images, an homography technique is employed using 4 known targets on a plane in an industrial environment (having various lights and contrasts). Using the camera intrinsic parameters, the homography matrix is computed then analyzed with a SVD to obtain the rotation matrix R and translation $\vec{t}$. The ground truth segmentation mask among the normalized depth are computed using the CAD model via OpenGL, placed according to the previous computed pose (Fig. 5). It is assumed that the homography matrix estimation is quite precise and suitable to establish a proper ground truth. The final dataset is composed with 100 real images covering a large portion of the object pose space (excluding z translations). A virtual images dataset is also build based on 500 random samples.

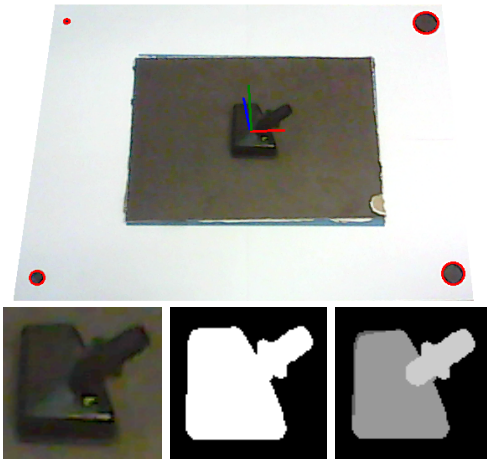

Figure 5. Pose estimation via homography using a blob detection (top) for patch, mask and depth ground truth creation using openGL (bottom).

The metric used for the segmentation evaluation is the mean intersection over union (mIoU) including 0.9 IoU. The orientation error can be expressed as a mean Euler angles error M_E or based on the *SO(3)* geodesic distance between to rotation matrix R_1 and R_2, $||log\left(R_1^T R_2\right)||$, adapted to the quaternion space as M_G with the predicted quaternion q and the truth $\hat{q}$:

$$M_G = 2 \arccos(|q.\hat{q}|) \tag{4}$$

The final pose estimation evaluation is done using several metrics including 2D-proj, 5cm5° and ADD.[17] The 2D-proj metric evaluates the pixels distances, between the truth object pixels and the CAD model projection (placed according to the estimated pose), that are inferior to 5px. The ADD metric quantifies the displacement, between the ground truth points cloud and the CAD model (again placed according to the estimated pose), that are inferior to 10% of the object diameter. All inferences are made using the results of the previous network inferences so that each error can be seen as a cumulative error within the pipeline.

7.2 Results

fdgdgfg

Table 1. Segmentation, depth, orientation and pose network performances for virtual images and real images.

Dataset	Segmentation	Depth	Orientation	x-y-Translations	**Final**
Rendered images	0.9 IoU mIOU	MSE	M_E M_G	MSE	2D-proj 5cm5° ADD
Real images	0.9 IoU mIOU	MSE	M_E M_G	MSE	2D-proj 5cm5° ADD

SSSS

Figure 6. Typical cases of good inferences (left) and badly orientated part (right).

8. CONCLUSION AND FUTURE WORKS

REFERENCES

[1] Wohlhart, P. and Lepetit, V., "Learning descriptors for object recognition and 3d pose estimation," *CVPR* (2015).

[2] Schwarz, M., Schulz, H., and Behnke, S., "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," *ICRA* (2015).

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

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

[5] Krull, A., Brachmann, E., Nowozin, S., Michel, F., Gumhold, S., and Rother, C., "Poseagent: Budget-constrained 6d object pose estimation via reinforcement learning.," *CVPR* (2017).

[6] Doumanoglou, A., Balntas, V., Kouskouridas, R., and Kim, T.-K., "Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation," *arXiv:1607.02257* (2016).

[7] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., "Going deeper with convolutions.," *CVPR* (2015).

[8] Kendall, A., Grimes, M., and Cipolla, R., "Posenet: A convolutional network for real-time 6-dof camera relocalization," *ICCV* (2015).

[9] Liu, W., Angualov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C.-Y., and Berg, A. C., "Ssd: single shot multibox detector.," *ECCV* (2016).

[10] Kehl, W., Manhardt, F., Tombari, F., Ilic, S., and Navab, N., "Ssd-6d: making rgb-based 3d detection and pose estimation great again.," *ICCV* (2017).

[11] Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D., "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes.," *RSS* (2018).

[12] Jafari, O. H., Mustikovela, S. K., Pertsch, K., Brachmann, E., and Rother, C., "ipose: Instance-aware 6d pose estimation of partly occluded objects," *arXiv:1712.01924* (2018).

[13] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P. V. D., Cremers, D., and Brox, T., "Flownet: Learning optical flow with convolutional networks," *Proceedings of the IEEE international conference on computer vision* , 2758–2766 (2015).

[14] Li, Y., Wang, G., Ji, X., Xiang, Y., and Fox, D., "Deepim: Deep iterative matching for 6d pose estimation," *ECCV* (2018).

[15] Karen, S. and Zisserman, A., "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556* (2014).

[16] Rad, M. and Lepetit, V., "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth.," *ICCV* (2017).

[17] Hinterstoisser, S., Ilic, V. L. S., Holzer, S., Bradski, G., Konolige, K., and Navab, N., "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," *ACCV* (2012).