End-to-end 6-DoF Object Pose Estimation through Differentiable Rasterization

Andrea Palazzi, Luca Bergamini, Simone Calderara, and Rita Cucchiara

University of Modena and Reggio Emilia, Italy
{name.surname}@unimore.it

Abstract. Here we introduce an approximated differentiable renderer to refine a 6-DoF pose prediction using only 2D alignment information. To this end, a two-branched convolutional encoder network is employed to jointly estimate the object class and its 6-DoF pose in the scene. We then propose a new formulation of an approximated differentiable renderer to re-project the 3D object on the image according to its predicted pose; in this way the alignment error between the observed and the re-projected object silhouette can be measured. Since the renderer is differentiable, it is possible to back-propagate through it to correct the estimated pose at test time in an online learning fashion. Eventually we show how to leverage the classification branch to profitably re-project a representative model of the predicted class (i.e. a medoid) instead. Each object in the scene is processed independently and novel viewpoints in which both objects arrangement and mutual pose are preserved can be rendered.

Differentiable renderer code is available at:
https://github.com/ndrplz/tensorflow-mesh-renderer.

Keywords: 6-DoF pose estimation · Differentiable rendering

1 Introduction

Inferring the six degrees of freedom (6-DoF) pose (3D rotations + 3D translations) of an object given a single RGB image is extremely challenging. Indeed, this process underlies a deep knowledge of the object itself and of the 3D world that is not easy to distill from a single frame; the kind of object, its 3D shape and the possible 3D transformation that leads to visually plausible outputs must be inferred jointly.
In this work, we show how an approximate differentiable renderer can be exploited to refine the 6-DoF pose estimation prediction using only 2D silhouette information. Keeping the object volume fixed we can back-propagate to the second renderer input, namely the object pose (see Fig. 1). We demonstrate that this differentiable block can be stacked on a 6-DoF pose estimator to significantly refine the estimated pose using only the 2D alignment information between the input object mask and the rendered silhouette. Leaving aside camera intrinsics,

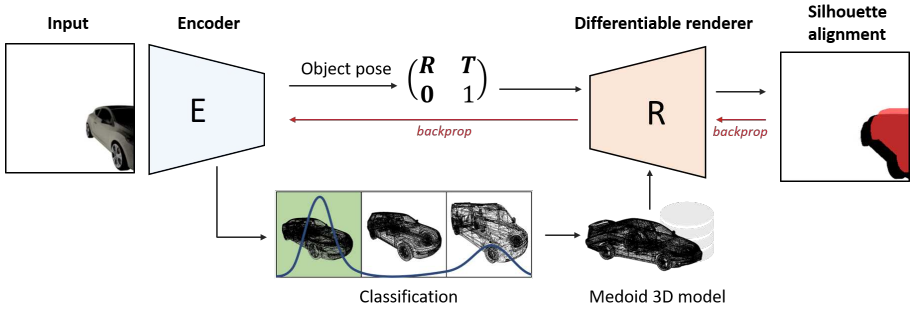


Fig. 1. The overall proposed framework. A deep convolutional encoder is fed with the object mask and predicts both the object's class and 6-DoF pose. By means of a differentiable renderer the predicted cluster medoid can be projected back according to the predicted pose, adding a further online alignment supervision w.r.t. the input mask.

a renderer can be generally thought as a black-box with two inputs and one output. The renderer takes as input (i) a given representation of the 3D object (e.g. voxels, mesh etc.) and (ii) the 6-DoF pose of the object w.r.t. the camera and produces the 2D image of the object or, as in our setting, solely its silhouette. Typically a rendering algorithm includes many non-differentiable operations (e.g. rounding, hard assignments etc.); it thus cannot be used in a deep learning architecture as it would break the back-propagation chain. Nonetheless, in the context of 3D volume estimation recent works [18,47,24,14,41] have been proposed which exploit approximated differentiable renderers to back-propagate the loss to the first renderer input, namely the 3D representation of the object, but leaving fixed the set of possible camera poses.

Since we rely on an fixed 3D model of the object we can abandon the redundant and expensive voxel representation in favor of meshes, which are lightweight and better tailored to represent 3D models [34]. Also, in contrast w.r.t. previous works, the rendering pipeline is implemented via a *rastering* algorithm, significantly faster than the conventional ray-tracing approach. Eventually, to solve the issue that true 3D model of the object is not usually known at test time, we indicate as a viable solution to perform coarse-grained classification and use a representative 3D model of the object category (*e.g.* a cluster medoid) instead. We experimentally demonstrate that the proposed pipeline is able to correct the estimated pose effectively even when using surrogate models.

2 Related Works

Beyond all doubt, the ImageNet [10] dataset has been the essential ingredient to many recent advances, since for the first time enough data were available for training very large (deep) models which in turn shook many benchmarks

[33,16,30,6]. More recently the large-scale database of synthetic models ShapeNet [5] dataset is having an analogous impact on the 3D community, showing that, in presence of enough data, 3D geometry and deep learning can be integrated successfully [37,46,7,36,22,2]. One of the areas in which this marriage is being fertile the most is the one of estimating the 3D shape of an object given an image, or to generate novel views of the same object.

Indeed, pre deep learning methods [4,15,20,42,29,35] often need multiple views at test time and rely on the assumption that descriptors can be matched across views [13,1], handling poorly self-occlusions, lack of texture [32] and large viewpoint changes [25]. Conversely, more recent works [37,46,7,36,22,2] are built upon powerful deep learning models trained on virtually infinite synthetic data rendered from ShapeNet [5]. From a high level perspective, we can distinguish methods that learn an implicit representation of object pose and volume and then decode it by means of another deep network [38,11,7,48] from methods that infer from the image a valid 3D representation (e.g. voxel-based) that can be re-projected by means of a differentiable renderer [47,41,14,44] to eventually measure its consistency w.r.t. the input image. Works leveraging the latter approach are strictly related to our proposed method in that they all found different ways to back-propagate through the renderer in order to correct the predicted object volume. Yan *et al* [47], Gadelha *et al* [14] and Wiles *et al* [44] take inspiration from the *spatial transformer network* [18] in the way the predicted volume is sampled to produce the output silhouette, even though they differ in the way the contribution of each voxel is counted for each line of sight. Rendering process proposed in Rezende *et al* [31] has to be trained via REINFORCE [45] since it is not differentiable. Tulsiani *et al*[41] frame the rendering phase in a probabilistic setting and define ray potential to enforce consistency. Our method differs substantially from all these works in several features. First, we keep the volume fixed and backpropagate through the renderer to correct the object pose, while the aforementioned works project the predicted 3D volume from a pre-defined set of poses (e.g. 24 azimuthal angles 0°, 15°, ... 345° around y-axis) and backpropagate the alignment error to correct the volume. Furthermore, while all these works use ray-tracing algorithm for rendering, our work is the first to propose a *differentiable raster-based renderer*. Eventually, all mentioned works represent the volume using voxels, which is inefficient and redundant since almost all valuable information is in the surface [34], while we use its natural parametrization by vertices and faces, i.e. the mesh. Convolutional neural networks (CNNs) have demonstrated analogous effectiveness in the task of object pose estimation, traditionally framed as a Perspective-n-Points (PnP) correspondence problem between the 3D world points and their 2D projections in the image [21,26]. With respect to descriptor-based methods [8,9,25], modern methods relying on CNNs [23,37,40] can solve ambiguities and handle occluded keypoints thanks to their high representational power and composite field of view, and have shown impressive results in specific tasks such as the one of human pose estimation [27,43,39,49]. Building upon this success, recent methods [50,28] combine CNN-extracted keypoints and deformable shape models in

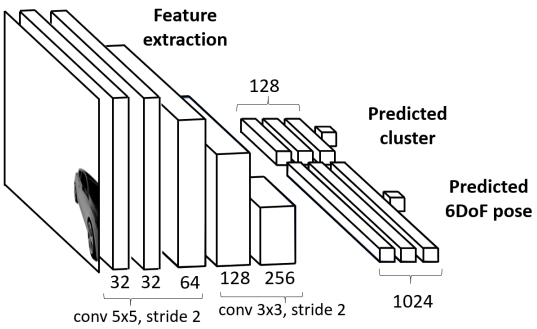


Fig. 2. Architecture of the encoder network. Visual features are extracted from the input image by means of 2D convolutions (first three layers have 5x5 kernel, last two have 3x3 kernel. All convolutional layers have stride 2 and are followed by leaky ReLu non-linearities). The flattened feature vector is fed to two fully connected branch, which estimate the object class and pose respectively.

a unique optimization framework to jointly estimate the object pose and shape. Differently from all these works, here we propose a substantially new method to integrate the object shape and pose estimation and model fitting in a unique end-to-end differentiable framework. To the best of our knowledge, this is the first work in which a differentiable renderer is used to correct the 6-DoF object pose estimation just by back-propagating 2D information on silhouette alignment error.

3 Model Description

Given a single RGB image in which one or more objects of interest has already been segmented, we train a deep convolutional encoder to predict the class and the 6-DoF pose (rotation and translation) of each object w.r.t the camera. We then exploit an approximate renderer to re-project the silhouette of object on the image according to the pose predicted by the encoder. As the true object models are not available at test time, for re-projection a representative object (*i.e.* medoid) of the predicted class is used. Also, since the rendering phase is approximated with a differentiable function, we can not only measure the alignment error w.r.t. the input object mask, but also back-propagate it to the encoder weights. Eventually, this allows us to fine-tune the encoder online optimizing just the alignment error. Our overall architecture is depicted in Fig. 1. In what follows both the encoder and the renderer models are detailed.

3.1 Encoder

The deep convolutional encoder network is schematized in Fig. 2. The first part of the network is dedicated to feature extraction and it is shared by the classification

and the pose estimation branch. The network has been designed inspired by [38] which showed favorable results in a related task. Features extracted are then used by two fully-connected independent branches to infer the object class and the camera pose respectively. All layers but the last are followed by leaky ReLu activation with $\alpha = 0.2$. Differently from most of the literature [47,14,44] we do not quantize the pose space into a discrete set of pre-defined poses to ease the task. Conversely, given a rotation matrix $\mathbf{R}_{3\times3}$ and a translation vector $\mathbf{t}_{3\times1}$ we regress the object pose

$$\mathbf{P}_{3\times4} = \begin{bmatrix}\mathbf{R}\ \mathbf{t}\end{bmatrix} \tag{1}$$

by optimizing the mean square error between the predicted and the true pose:

$$\mathcal{L}(\mathcal{X}, \mathcal{Y}_p, \theta) = \frac{1}{N}\sum_i ||y_i - f_p(x_i, \theta)||^2 \quad x_i \in \mathcal{X}, y_i \in \mathcal{Y}_p \tag{2}$$

where $\mathcal{X}$ is the set of RGB images, $\mathcal{Y}_p$ is the set of true $\mathbf{P}_{3\times4}$ pose matrices and $f_p(x_i, \theta)$ is the pose predicted by the encoder for example x_i according to its weights θ. From a technical standpoint, for each X, Y, Z axis the encoder regresses the cosine of the Euler rotation angle and the respective translation. The output roto-translation matrix is then composed following Euler ZYX convention: in this way predicted matrices are guaranteed to be always geometrically consistent. For the classification branch we instead optimize the following categorical cross-entropy function:

$$\mathcal{L}(\mathcal{X}, \mathcal{Y}_c, \theta) = -\frac{1}{N}\sum_i y_i \log f_c(x_i, \theta) \tag{3}$$

where $x_i \in \mathcal{X}$ is an input RGB image, $f_c(x_i, \theta)$ is the encoder predicted distribution over possible clusters for example x_i and y_i in the true one-hot distribution for example x_i.

3.2 Differentiable Renderer

To measure the reliability of the predicted 6-DoF pose and to be able to correct it at test time, we design a fully differentiable renderer for re-projecting the silhouette of the 3D model on the image according to the predicted object pose. This allows to refine the estimated pose by back-propagating the alignment error between the 2D silhouettes. To the best of our knowledge, it is the first time that a fully-differentiable raster-based renderer is used to this purpose. Differently from concurrent works such as [47], our rendering process starts from the raw mesh triangles and not from a 3D voxel representation. While the latter is easier to predict by a neural network since it has a static shape, its footprint scales with the cube of the resolution and forces to use ray-tracing techniques to render the final image, known to be slow and harder to parallelize. Despite rastering does not allow for photo-realistic shaded images, as it does not imply light sources rays tracing, it is still well suited for all tasks which require the object shape silhouette from different point of views as in our case.
Our renderer is composed of two main parts:

– A *rastering algorithm*, which applies the predicted camera to the 3D triangles meshes to obtain 2D projected floating point coordinates of the corners;
– An *in/out test* to determine which projected points lie inside the triangles, *i.e.* which triangles must be filled.

While the first step is fully differentiable, a naive implementation of the latter exploits boolean masks to select the pixels to be filled, which eventually breaks the backpropagation through the network. Inspired by [18], we employed a spatial transformation to assign a value to each pixel based on a relation between its coordinates and those of the triangles corners. While a boolean mask represents hard membership, this approach assigns each pixels a continuous value, thus applying a soft (differentiable) membership. From a more technical standpoint, given all triangles T which compose the mesh of current model, we project the 3D triangle vertices V_{3D} as follows:

$$\begin{bmatrix} V_{2D} \\ 1 \end{bmatrix} = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} = \mathbf{K}_{3\times3}\mathbf{P}^{-1}_{3\times4} \begin{bmatrix} V_{3D} \\ 1 \end{bmatrix} \tag{4}$$

where $\mathbf{K}_{3x3}$ is the camera calibration matrix and $\mathbf{P}^{-1}_{3\times4}$.

Then, defined as $E^{(i)} = [(v_1, v_0), (v_2, v_1), (v_0, v_2)]$ the three edges of the *i-th* projected triangle, the renderer's output for pixel in location (u, v) can be computed as:

$$g_{u,v} = \sum_{i}^{T} F_{norm} \left(\prod_{(v_j, v_k)}^{E^{(i)}} \max \left(\begin{vmatrix} v_j - v_k \\ v_j - (u, v) \end{vmatrix} \begin{vmatrix} v_1 - v_0 \\ v_2 - v_1 \end{vmatrix}, 0 \right) \right), \quad (u, v) \in H \times W$$

$$where \quad F_{norm}(x) = \tanh \frac{x - min(x)}{max(x) - min(x)} \tag{5}$$

and H, W indicate the image height and width in pixels. We refer the reader to Fig. 3 for a better intuition of Equation 5. It is worth noticing that the *i-th*

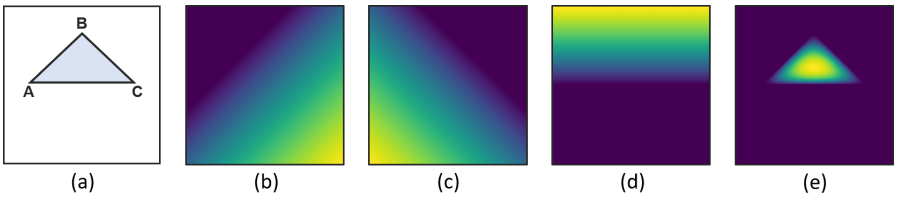


Fig. 3. Exemplification of the approximated rastering process. First each triangle composing the mesh is projected in the 2D image (a) using Eq. 4. The determinant product inside the max of Eq. 5 selects the points which lie on the left side of each edge of the triangle (b), (c), (d). The product of these three terms gives an approximated yet differentiable rendering of the triangle's silhouette (e).

triangle contributes to the output only if all the three determinant products are positive, meaning that (u, v) point lies on the left side of all three triangle edges *i.e.* it is inside the triangle.

4 Experiments

4.1 Dataset

We train our model on ShapeNetCore(v2) [5] dataset, which comprises more than 50K unique 3D models from 55 distinct man-made objects. We focus in particular on the car synset since it is one of the most populated category with 7497 different 3D CAD vehicle models. Each model is stored in .obj format along with its materials and textures: dimensions, number of vertices and details vary greatly from one model another.

Data collection To collect the data, we first load a random model on the origin $\mathbf{t} = (0, 0, 0)$ of our reference system. We then create a camera in location $\mathbf{t} = (x, y, z)$. While on xy plane the location is randomly sampled in a $q_x \times q_y$ grid, we keep fixed $z = k$ under the assumption that the camera is mounted somewhere at height k on a moving agent (e.g. an unmanned vehicle). We then force the camera to point an empty object e that is randomly sampled at $z = 0$ and x, y sampled as above in a $e_x \times e_y$ grid: in this way we make the object to appear translated in the camera image. Eventually, the camera image is dumped along with the camera pose to constitute an example x_i. We refer the reader to Fig. 4 to get a better insight into the procedure. *Data collection details:* In our experiments we set $q_x = q_y = 10$ and $k = 1.5$, which is the average height of a European vehicle. For the empty object we set $e_x = e_y = 3$. Models are standardized s.t. the major dimension has length 6. For each cluster, the models are split with ratio 0.6-0.2-0.2 into train, validation and test set respectively. Medoids are expected to be known at test-time and do not belong to any of the splits. Models are rendered using Blender CYCLES engine [3] to maximize photo-realism.

Selecting the representative 3D model Since the true 3D object model is hardly available at test time, we want to verify if a surrogate 3D model can be instead successfully employed for the rendering process. Analogously to Du *et al* [12] we distinguish three main vehicle clusters, namely i) *Sedan* passenger cars, ii) *Sport-utility vehicles* (SUV, which are also passenger cars but have off-road features like raised ground clearance) and iii) *Cargo* vehicles such as trucks and ambulances. Aligned CAD models for the three clusters are depicted in Fig. 4(c). Following Tatarchenko *et al* [38] we selected the representative model for each cluster, by extracting and comparing the HOG descriptors from two standard rendered views of each CAD model (i.e. frontal and side). Eventually we compute the $L2$ distance between descriptors and for each cluster we retain the cluster medoid, *i.e.* the model with the least average distance from all the others.

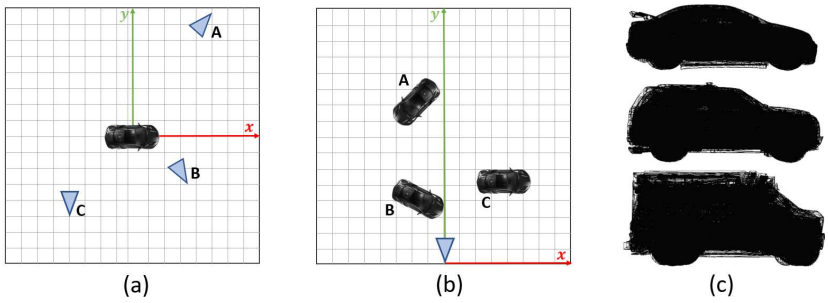


Fig. 4. On the left is depicted how all camera poses predicted by the encoder independently for each object (a) can be roto-translated to a common origin to reconstruct the overall scene (b), also in Fig. 7. On the right, the average silhouette of vehicles belonging to *sedan*, *SUV* and *cargo* is shown (c). For each cluster all 3D meshes are overlaid before taking the snapshot from the side view; the high overlap highlights the low intra-cluster variance.

4.2 Model Evaluation

Metrics The encoder ability to estimate the 3D pose of the object is measured by means of geodesic distance between predicted and true rotation matrix [40,17] as:

$$\Delta(\mathbf{R}_{true}, \mathbf{R}_{pred}) = \frac{||\log(\mathbf{R}_{true}^T \mathbf{R}_{pred})||_F}{\sqrt{2}} \tag{6}$$

where $||\mathbf{A}||_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$ indicates the Frobenius norm. In particular, we report the median value of the aforementioned distance over all predictions in test set as Median Viewpoint Error (MVE). We also report the percentage of examples in which the pose rotation error is smaller than $\pi/6$ as $Acc_{\frac{\pi}{6}}$. To measure the re-projection alignment error we instead rely on mean intersection over union (mIoU) metric defined over the N test examples as $\frac{1}{N}\sum_i \frac{S_i \cap \tilde{S}_i}{S_i \cup \tilde{S}_i} \quad i = 1, \ldots, N$: where S_i is the ground truth silhouette and $\tilde{S}_i = g(f_p(x_i), f_c(x_i), \mathbf{K})$ is the renderer output given the predicted object pose, cluster and camera intrinsics $\mathbf{K}$.

Model performance To prove the effectiveness of the proposed method we first train the 6-DoF pose estimation network alone to jointly estimate the object class and its 6-DoF pose. In this way, we get a baseline to measure the successive contribute of the prediction refinement through our differentiable rendering module. State-of-the-art results on test set reported in Table 1(first row) indicate this to be already a strong baseline. The prediction refinement module is then plugged-in, and the evaluation is repeated. For each example, the medoid of the predicted class is rendered according to the predicted pose, back-propagating the alignment error between the true and the rendered silhouette for 30 optimization steps. Results of this analysis are reported in Table 1(second row) and indicate a huge performance gain (20%) obtainable by maximizing the 2D alignment between object masks.

Table 1. Table summarizing model performance. It is worth noticing that none of the metrics in the table are explicitly optimized during refinement. Results of concurrent works on the vehicle class are shown for reference, despite the task of [40,37] is only viewpoint estimation (not 6-DoF pose) and all are trained on different dataset.

Model	**Accuracy** ↑	**mIoU** ↑	**MVE** ↓	**Acc**$_{\frac{\pi}{6}}$ ↑
encoder	0.89	0.59	5.7	0.86
encoder+refinement	**0.89**	**0.72**	**4.5**	**0.90**
Pavlakos *et al* [28]	-	-	6.9	-
Tulsiani and Malik [40]	-	-	9.1	0.89
Su *et al* [37]	-	-	6.0	0.88

The significant improvement in all the metrics, despite none of these is optimized explicitly, suggests that the proposed differentiable rendering module is a viable solution for refining the predicted 6-DoF even at test time, requiring minimal information (*i.e.* only the object mask). The process of prediction refinement can be appreciated in Fig. 5.

Renderer ablation study We measure, at first, the impact of rendering resolution on the optimization process by refining the object 6-DoF estimated pose using different rendering resolutions. Results reported in Table 2 show that working at higher resolution is definitely helpful while very-low resolution are hardly beneficial, if not detrimental, for the optimization process. This supports the need to abandon the voxel-based representation, whose computational footprint increases with the cube of resolution. We then compare our renderer with the publicly available implementation of Perspective Transformer Network (PTN) by Yan *et al*[47]. Results are shown in Fig. 6(a). Since PTN relies on a fixed 32x32x32 voxel representation, rendering at higher resolution hardly changes the output's fidelity w.r.t. the true silhouette. Conversely, our mesh-based renderer is able to effectively take advantage of the higher resolution. Comparing

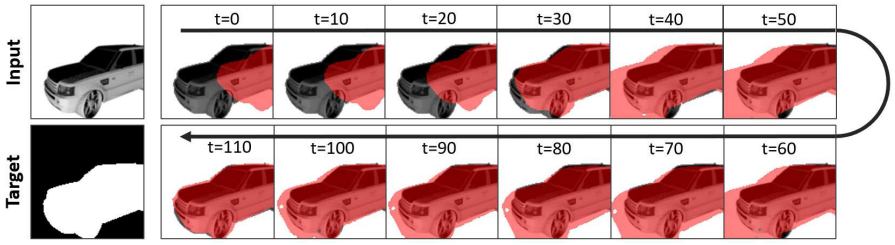
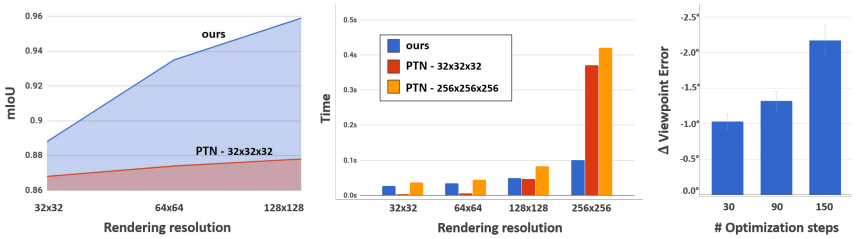


Fig. 5. Online refinement of the estimated pose; We overlay in red the predicted silhouette for each optimization step. Despite the initial estimate (t=0) was noticeably wrong, the 6-DoF object pose is gradually corrected using only 2D silhouette alignment information.

Table 2. Gains obtained in pose estimation using different rendering resolutions. Increasing the resolution used for rendering the silhouette is much beneficial to the optimization process. Conversely, for very low resolution this phase is hardly helpful.

Renderer Resolution	Δ IoU ↑	Δ Viewpoint Error ↓	Δ Translation Error ↓
16x16	+0.00	+0.15	+0.02
32x32	+0.03	-0.26	+0.00
64x64	+0.05	-0.57	+0.00
128x128	+0.11	-1.03	-0.01
256x256	+0.13	-1.29	-0.03



(a) (b) (c)

Fig. 6. (a) Intersection over union between rendered silhouette and the ground truth one for both our renderer and Perspective Transformer Networks (PTN) [47], at different rendering resolutions. (b) Rendering time for different image (and PTN voxel) resolutions. (c) Average viewpoint error improvement for different number of optimization steps. See text for details.

our rendering time with PTN [47] in Fig. 6(b), we see that PTN scores favorably only for very-low voxel and image resolutions, while as resolution increases the PTN rendering time increases exponentially due to the voxel-based representation. Eventually, in Fig. 6(c) we show that our average viewpoint error continues to decrease along with the number of refinement optimization steps.

Training details Encoder is trained until convergence with batch size=64 and ADAM optimizer with learning rate 10^{-5} (other hyper-parameters as suggested in the original paper [19]). Batch size is decreased to 20 and learning rate to 10^{-6} during renderer fine-tuning. We find useful dropout ($p = 0.5$) after all dense layers and $L2$ weight decay over feature extraction for regularization purposes.

5 Conclusions

In this work we introduce a 6-DoF pose estimation framework which allows an *online* refinement of the predicted pose from minimal 2D information (*i.e.* the object mask). A fully differentiable raster-based renderer is developed for re-projecting the object silhouette on the image according to the predicted 6-DoF pose: this allows to correct the predicted pose by simply back-propagating the

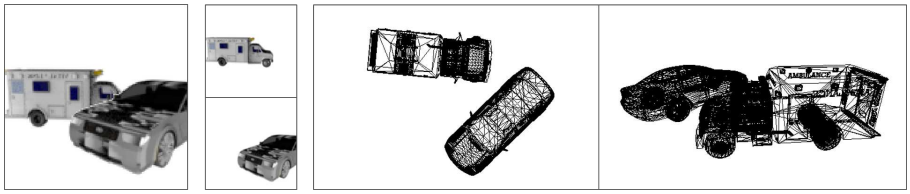


Fig. 7. Qualitative results for multiple object scenes. Since all predicted poses lie in the same reference system (see Fig. 4), different views of the scene can be produced by means of any rendering engine. It is worth noticing that each object has been substituted by the representative model for its predicted class.

alignment error between the observed and the rendered silhouette. Experimental results indicate i) the overall effectiveness of the online optimization phase, ii) that proxy representative models can be profitably used in place of the true ones in case these are not available and iii) the benefit of working in higher resolution, well-handled by our raster-based renderer but hardly managed by concurrent ray-tracing, voxel-based algorithms.

References

1. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building rome in a day. Communications of the ACM **54**(10), 105–112 (2011)
2. Aubry, M., Maturana, D., Efros, A.A., Russell, B.C., Sivic, J.: Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3762–3769 (2014)
3. Blender Online Community: Blender - a 3D modelling and rendering package. Blender Foundation, Blender Institute, Amsterdam (2017), http://www.blender.org
4. Boyer, E., Franco, J.S.: A hybrid approach for computing visual hulls of complex objects. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 695–701. IEEE Computer Society Press (2003)
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
6. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2018)
7. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: European Conference on Computer Vision. pp. 628–644. Springer (2016)
8. Collet, A., Berenson, D., Srinivasa, S.S., Ferguson, D.: Object recognition and full pose registration from a single image for robotic manipulation. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. pp. 48–55. IEEE (2009)

9. Collet, A., Martinez, M., Srinivasa, S.S.: The moped framework: Object recognition and pose estimation for manipulation. The International Journal of Robotics Research **30**(10), 1284–1306 (2011)
10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 248–255. IEEE (2009)
11. Dosovitskiy, A., Springenberg, J.T., Tatarchenko, M., Brox, T.: Learning to generate chairs, tables and cars with convolutional networks. IEEE transactions on pattern analysis and machine intelligence **39**(4), 692–705 (2017)
12. Du, X., Ang Jr, M.H., Karaman, S., Rus, D.: A general pipeline for 3d detection of vehicles. ICRA (2018)
13. Fitzgibbon, A., Zisserman, A.: Automatic 3d model acquisition and generation of new images from video sequences. In: Signal Processing Conference (EUSIPCO 1998), 9th European. pp. 1–8. IEEE (1998)
14. Gadelha, M., Maji, S., Wang, R.: 3d shape induction from 2d views of multiple objects. 3D Vision (2017)
15. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 43–54. ACM (1996)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
17. Huynh, D.Q.: Metrics for 3d rotations: Comparison and analysis. Journal of Mathematical Imaging and Vision **35**(2), 155–164 (2009)
18. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in neural information processing systems. pp. 2017–2025 (2015)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Kolev, K., Klodt, M., Brox, T., Cremers, D.: Continuous global optimization in multiview 3d reconstruction. International Journal of Computer Vision **84**(1), 80–96 (2009)
21. Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnp: An accurate o (n) solution to the pnp problem. International journal of computer vision **81**(2), 155 (2009)
22. Lim, J.J., Khosla, A., Torralba, A.: Fpm: Fine pose parts-based model with 3d cad models. In: European Conference on Computer Vision. pp. 478–493. Springer (2014)
23. Long, J.L., Zhang, N., Darrell, T.: Do convnets learn correspondence? In: Advances in Neural Information Processing Systems. pp. 1601–1609 (2014)
24. Loper, M.M., Black, M.J.: Opendr: An approximate differentiable renderer. In: European Conference on Computer Vision. pp. 154–169. Springer (2014)
25. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
26. Moreno-Noguer, F., Lepetit, V., Fua, P.: Accurate non-iterative o (n) solution to the pnp problem. In: Computer vision, 2007. ICCV 2007. IEEE 11th international conference on. pp. 1–8. IEEE (2007)
27. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision. pp. 483–499. Springer (2016)
28. Pavlakos, G., Zhou, X., Chan, A., Derpanis, K.G., Daniilidis, K.: 6-dof object pose from semantic keypoints. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on. pp. 2011–2018. IEEE (2017)

29. Pollefeys, M., Koch, R., Vergauwen, M., Van Gool, L.: Metric 3d surface reconstruction from uncalibrated image sequences. In: European Workshop on 3D Structure from Multiple Images of Large-Scale Environments. pp. 139–154. Springer (1998)
30. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
31. Rezende, D.J., Eslami, S.A., Mohamed, S., Battaglia, P., Jaderberg, M., Heess, N.: Unsupervised learning of 3d structure from images. In: Advances In Neural Information Processing Systems. pp. 4996–5004 (2016)
32. Saponaro, P., Sorensen, S., Rhein, S., Mahoney, A.R., Kambhamettu, C.: Reconstruction of textureless regions using structure from motion and image-based interpolation. In: Image Processing (ICIP), 2014 IEEE International Conference on. pp. 1847–1851. IEEE (2014)
33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
34. Sinha, A., Unmesh, A., Huang, Q., Ramani, K.: Surfnet: Generating 3d shape surfaces using deep residual networks. In: Proc. CVPR (2017)
35. Starck, J., Hilton, A.: Model-based human shape reconstruction from multiple views. Computer Vision and Image Understanding **111**(2), 179–194 (2008)
36. Stark, M., Goesele, M., Schiele, B.: Back to the future: Learning shape models from 3d cad data. In: Bmvc. vol. 2, p. 5. Citeseer (2010)
37. Su, H., Qi, C.R., Li, Y., Guibas, L.J.: Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2686–2694 (2015)
38. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Multi-view 3d models from single images with a convolutional network. In: European Conference on Computer Vision. pp. 322–337. Springer (2016)
39. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1653–1660 (2014)
40. Tulsiani, S., Malik, J.: Viewpoints and keypoints. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1510–1519 (2015)
41. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: CVPR. vol. 1, p. 3 (2017)
42. Vogiatzis, G., Esteban, C.H., Torr, P.H., Cipolla, R.: Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(12), 2241–2246 (2007)
43. Wei, S.E., Ramakrishna, V., Kanade, T., Sheikh, Y.: Convolutional pose machines. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4724–4732 (2016)
44. Wiles, O., Zisserman, A.: Silnet: Single-and multi-view reconstruction by learning from silhouettes. British Machine Vision Conference (2017)
45. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. In: Reinforcement Learning, pp. 5–32. Springer (1992)
46. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in Neural Information Processing Systems. pp. 82–90 (2016)
47. Yan, X., Yang, J., Yumer, E., Guo, Y., Lee, H.: Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In: Advances in Neural Information Processing Systems. pp. 1696–1704 (2016)

48. Yang, J., Reed, S.E., Yang, M.H., Lee, H.: Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In: Advances in Neural Information Processing Systems. pp. 1099–1107 (2015)
49. Zhou, X., Zhu, M., Leonardos, S., Derpanis, K.G., Daniilidis, K.: Sparseness meets deepness: 3d human pose estimation from monocular video. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4966–4975 (2016)
50. Zhu, M., Zhou, X., Daniilidis, K.: Single image pop-up from discriminatively learned parts. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 927–935 (2015)