IEEE-RAS International Conference on Humanoid Robots (Humanoids), Toronto, Canada, October 2019.

Refining 6D Object Pose Predictions using Abstract Render-and-Compare

Arul Selvam Periyasamy*, Max Schwarz*, and Sven Behnke

***Abstract*— Robotic systems often require precise scene analysis capabilities, especially in unstructured, cluttered situations, as occurring in human-made environments. While current deep-learning based methods yield good estimates of object poses, they often struggle with large amounts of occlusion and do not take inter-object effects into account. Vision as inverse graphics is a promising concept for detailed scene analysis. A key element for this idea is a method for inferring scene parameter updates from the rasterized 2D scene. However, the rasterization process is notoriously difficult to invert, both due to the projection and occlusion process, but also due to secondary effects such as lighting or reflections. We propose to remove the latter from the process by mapping the rasterized image into an abstract feature space learned in a self-supervised way from pixel correspondences. Using only a light-weight inverse rendering module, this allows us to refine 6D object pose estimations in highly cluttered scenes by optimizing a simple pixel-wise difference in the abstract image representation. We evaluate our approach on the challenging YCB-Video dataset, where it yields large improvements and demonstrates a large basin of attraction towards the correct object poses.**

I. Introduction

Robust robotic interaction in environments made for humans is an open research field. An important prerequisite in this context is scene perception, yielding the necessary information such as detected objects and their poses or affordances for later manipulation actions. While there are various high-accuracy methods for scene understanding, the problem becomes significantly harder in the presence of clutter and inter-object effects. As such, current works in humanoid manipulation that require precise grasping are often limited to non-cluttered or even isolated scenes (e.g. [1], [2]). While the manipulation action itself and planning for it is certainly more difficult in cluttered scenes, robust 6D object pose estimation is a necessary prerequisite.

An interesting approach in this context is the idea of viewing computer vision as an inverse graphics process [3], [4]. It promises to perform scene analysis by inverting the rasterization process, which sounds highly promising—today's rendering techniques are capable of producing convincing photo-realistic renderings of highly complicated scenes, so inversion of the process should yield high-quality scene analysis. However, the problem plaguing the inverse graphics field is that the rendering process is largely unidirectional, with complex physical effects such as lighting, surface scattering, transparency, and so on. Furthermore, scene analysis

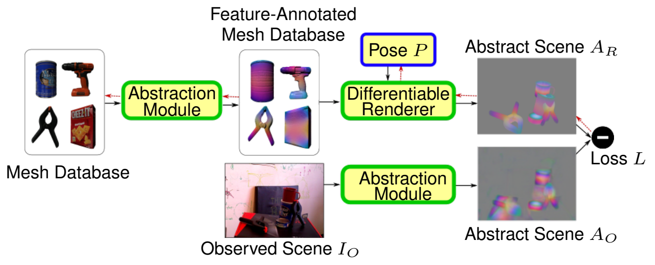

Fig. 1. Scene analysis using differential rendering and learned abstraction module. An abstract representation is extracted both for the input scene and the mesh database. The differential rendering module then tries to match the abstract scene representation with the feature-annotated meshes. The loss gradient (red dotted line) only needs to be backpropagated through the differentiable renderer.

is especially in demand for cluttered scenes, e.g. in warehouse automation contexts, but occlusion effects caused by clutter are among the most difficult to invert or differentiate.

To take a step towards a solution of this problem, we propose to first remove most *secondary* rendering effects from the scene using abstract surface features learned in an unsupervised manner. This way, only the *primary* effects remain—occlusion and projection. These effects can then be explained and analyzed by a simpler differentiable rendering component.

We apply our render-and-compare framework to the task of monocular 6D pose estimation, specifically pose refinement, where initial pose guesses are available. In our approach, 6D pose predictions from state-of-the-art pose estimation methods are refined by minimizing the pixelwise difference between the rendered image and the observed image in the proposed abstract descriptor space invariant to secondary rendering effects. We further make the assumption that meshes of the objects are available, as is the case in many industrial and robotic applications. For example, service robots operating in human environments working with tools designed for human usage and industrial part handling robots can greatly benefit from having precise 6D pose estimation.

In short, our contributions proposed in this work include:

1) A scene abstraction method that removes secondary render effects, so that scene analysis by render-and-compare becomes feasible,
2) fusion of surface features onto object meshes for direct rendering of scenes in the abstract feature space,
3) a fast and light-weight differentiable rendering component, and
4) the integration of these components into a pose refine-

*: The authors contributed equally.
All authors are with the Autonomous Intelligent Systems (AIS) Group, Computer Science Institute VI, University of Bonn, Germany.
Email: `periyasa@ais.uni-bonn.de`

ment pipeline, which is evaluated on the YCB Video Dataset [5]. To the best of our knowledge, this is the first differentiable rendering pipeline capable of optimizing object poses in cluttered real-world scenes.

This paper is structured as follows. After discussing related works in the section II, we describe the descriptor learning process in section III and introduce our differentiable renderer, LightDR in section IV. In section V, we apply our pipeline to 6D object pose refinement, and further analyze the robustness of the proposed refinement process.

II. Related Work

Vision as inverse graphics aims at inferring object parameters like shape, illumination, reflectance, and pose, scene parameters like camera parameters, lighting, and secondary reflections by inverting the rendering process. The process of rendering 3D scene to discrete 2D pixels involves discretization steps that are not differentiable. However, several approximation methods have been to proposed to realize a differentiable renderer. Loper and Black [6] proposed OpenDR, a generic differentiable renderer that can compute gradients with respect to object and scene parameters. Kato *et al.* [7] introduced a differentiable renderer that is suited for neural networks. Rezende *et al.* [8] treat forward rendering as a black-box and used REINFORCE [9] to compute gradients. Li *et al.* [10] proposed edge sampling algorithm to differentiate ray tracing that can handle secondary effects such as shadows or global illumination. Liu *et al.* [11] proposed a differentiable probabilistic formulation instead of discrete rasterization. The differentiable renderer used in this work is closely modeled after OpenDR but tailored for object pose refinement with a strong focus on speed.

Vision as inverse graphics is most often formulated as a render-and-compare approach, where model parameters are optimized by minimizing the difference between rendered and observed images. Zienkiewicz *et al.* [12] used render-and-compare for real-time height mapping fusion. Several recent works used render-and-compare for solving a wide range of vision problems: Tewari *et al.* [13] learned unsupervised monocular face reconstruction; Kundu *et al.* [14] introduced a framework for instance-level 3D scene understanding; Moreno *et al.* [15] estimated 6D object pose in cluttered synthetic scenes. More closely related is the DeepIM method by Li *et al.* [16], who formulated 6D object pose estimation as an iterative pose refinement process that refines the initial pose by trying to match the rendered image with the observed image. In contrast to our approach, they avoid the need for backpropagating gradients through the renderer by training a neural network to output pose updates. While the method yields very promising results, it is not directly clear how to apply this method to symmetric objects without specifying symmetry axes, whereas our method inherently optimizes to a suitable pose. We also note that DeepIM is object-centric, refining each object's pose separately. In contrast, our method retains the entire scene, refining all object poses simultaneously and thus is able to account for inter-object effects.

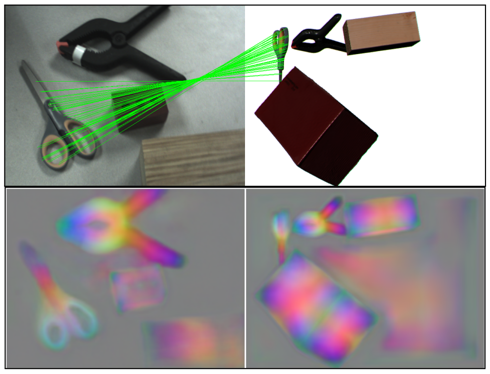
Fig. 2. Learning dense descriptors from real-synthetic correspondences. Top: Real and synthetic input frames A and B. Positive correspondences matches are shown in green for one object. Bottom: Learned dense abstract representation of the scene.

In this work, we use render-and-compare to refine 6D object poses in cluttered real-world scenes. Instead of comparing the rendered and observed RGB images, we propose to use an abstraction network to deal with the difficulties in comparing images from two different modalities.

III. Learned Descriptors for Scene Abstraction

Real scenes exhibit a large variety of secondary effects such as lighting, camera noise, reflections, and so on. All of these effects are very difficult to model and severely constrain the applicability of differentiable rendering methods. We propose an additional abstraction module $f : \mathbb{I} \to \mathbb{A}$, mapping the RGB image space I to an abstract feature space A. Ideally, the mentioned secondary effects lie in the null space of f. For convenience, we require that A is also image-like so that pixels in I correspond to feature vectors in A.

The very difficult problem of decomposing an image into its different intrinsic components, such as shading, reflectance, and shape, has been studied extensively [17]–[19]. However, in this application such a complex and physically accurate decomposition is not required. In order to be usable for differential rendering, we only require that features are similar for corresponding points on the same object (under varying lighting conditions etc), and dissimilar for non-corresponding objects. Many traditional feature extractors exhibit this property (e.g. SIFT). Of particular interest, however, are feature extractors designed for dense output. Recently, [20] showed that highly precise feature extractors can be trained in a self-supervised way from ground truth correspondences. The learned descriptors outperform sparse feature extractors by a large margin.

A. Real-Synthetic Correspondences for Descriptor Learning

In order to leverage this idea in the differential rendering setting, we propose to learn descriptors from the object meshes in conjunction with a training dataset for pose estimation. For a pose-annotated real dataset frame A, we render a synthetic frame B with the same object set, but using different poses (see Fig. 2). Corresponding points in both RGB frames can be easily determined from the object poses through projective geometry.

With a probability of 0.5, the synthetic object is drawn in the same orientation as the real object (see Fig. 2). This results in a large number of positive correspondences. In the other case, the orientation is drawn uniformly random—resulting in a larger number of negative examples and ensuring that the learned descriptors stay globally unique. The object translation is always sampled uniformly at random.

B. Network Details & Training

Since our application depends on high spatial resolution of the computed features, we follow the architecture of Light-Weight RefineNet [21], a state-of-the-art semantic segmentation method, which successively upsamples lower-resolution feature maps of higher abstraction levels and combines them with higher-resolution feature maps of lower abstraction levels. In our network, we use ResNet-34 [22] with ImageNet pretraining as the RefineNet backbone network. The final convolutional layer is adapted to not only output semantic segmentation ($C \times H \times W$) but also the dense features ($D \times H \times W$). Here, semantic segmentation is included as an auxiliary task, as it is not used in the following pipeline stages (but could be in the future). In our experiments, we use $D = 3$ for easier visualization, as in [23].

Following [20] and [23] we minimize a pixel-wise contrastive loss function $\mathcal{L}_C(A, B)$:

$$\mathcal{L}_+(A, B) = \frac{1}{|M_+|} \sum_{u \in M_+} ||f_A(u_A) - f_B(u_B)||_2^2 \quad (1)$$

$$\mathcal{L}_-(A, B) = \frac{1}{H_-} \sum_{u \in M_-} \max(0, M - ||f_A(u_A) - f_B(u_B)||_2)^2 \quad (2)$$

$$\mathcal{L}_C(A, B) = \mathcal{L}_+(A, B) + \mathcal{L}_-(A, B), \quad (3)$$

where $f_P(u)$ is the descriptor value in image P at location u, M_+ is the set of correspondent pixel pairs (u_A, u_B), M_- is a set of randomly sampled negative correspondences (also limited to the object mask), and $H_- = \sum_{u \in M_-} \mathbb{1}[M - ||f_A(u_A) - f_B(u_B)||_2 > 0]$ the number of *hard negatives*.

To encourage the network to disregard clutter in the background, we also introduce a loss on background pixels for a single frame F:

$$\mathcal{L}_{\text{bg}}(F) = \lambda \frac{1}{G} \sum_{u \in G} ||f_F(u)||_2^2, \quad (4)$$

where G is the set of background pixels. The loss balancing factor $\lambda = 0.1$ is chosen rather small in order not to hurt descriptor learning in the foreground pixels.

The combined loss function is simply

$$\mathcal{L}(A, B) = \mathcal{L}_C(A, B) + \sum_{F \in \{A,B\}} \mathcal{L}_S(F) + \mathcal{L}_{\text{bg}}(F), \quad (5)$$

where $\mathcal{L}_S$ is the cross-entropy loss for pixel-wise segmentation.

The network is trained using the Adam optimizer with learning rate 1e-4 and parameters $\beta_1 = 0.9, \beta_2 = 0.999$ on 350k image pairs. Note that the synthetic image B of each image pair is generated on the fly, i.e. the network never is presented with the same pair twice.

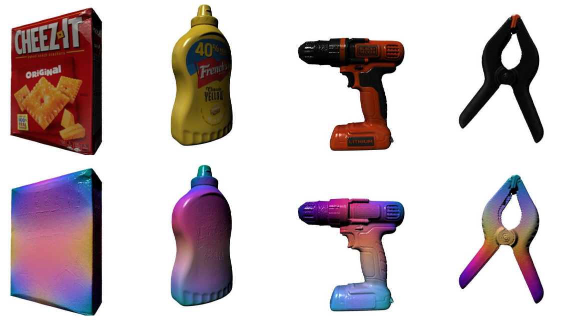
Fig. 3. Learned surface features, projected and fused onto the mesh. The 3D feature vectors are visualized directly as RGB colors.

C. Mesh Representation for Surface Features

Since the learned features should be constant for each local surface patch, independent of the viewing pose, we can *fuse* the descriptor information onto the mesh representation. To this end, we render N views (50 in our experiments) of the object from randomly sampled viewing directions and viewing distances. After feature extraction using the learned network, the resulting point-feature pairs are aggregated in the object frame. A voxel grid downsampling is applied, where descriptors and positions are each averaged inside each voxel. The voxel size is determined heuristically from the object bounding box s.t. the voxel count is constant—this results in constant-size output. This method is robust and easy to tune in case more complex geometry needs to be supported. In our experiments, we use 5000 voxels. Finally, each vertex of the object mesh is assigned an interpolated descriptor from the four nearest voxels using inverse-distance weighting. Figure 3 shows exemplary meshes and corresponding feature visualizations.

IV. Differentiable Renderer

The pose refinement problem is an optimization problem. In our case, we assume that we start with a pose initialization of reasonable quality, such that local optimization methods can find the optimum solution. In this context, it is very favorable to be able to compute derivatives of the rendering process, since the number of parameters grows linearly with the number of objects in the scene (at least six parameters per object). Optimization without gradient information quickly becomes infeasibly slow.

We base our differential rendering module on the method of OpenDR [6], which is able to approximate gradients with respect to lighting parameters, camera parameters, object poses, etc. We note that in our setting, only pose parameters need to be optimized, because lighting and other surface effects are removed by the abstraction network and camera parameters are assumed to be fixed in monocular pose estimation. Encouraged by this simplification, we implemented a lightweight differentiable renderer, which we call *LightDR*.

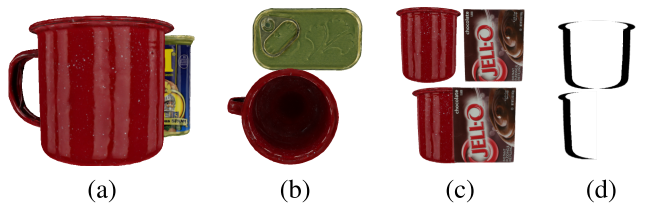

Fig. 4. Corner cases for render gradient estimation. (a) and (b): Front and top view of an exemplary scene with occlusion. (c) Top: Occlusion-free scene; Bottom: Scene with mug occluded. (d) Magnitude of pixel-wise loss corresponding to scenarios depicted in (c) when the mug is translated. Black depicts: high loss magnitude. When the rendered mug is moved behind the occluder, all pixels with high loss (and thus gradient information) lie outside of the rendered object mask.

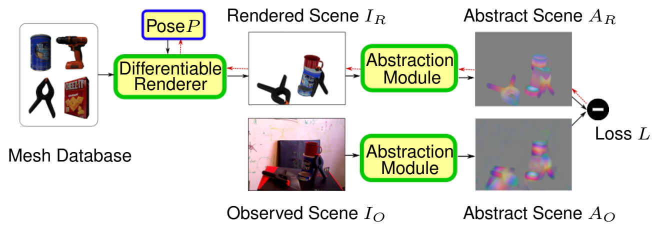

Fig. 5. Render-Abstraction pipeline. The renderer produces an RGB image of the scene, which is then mapped into the abstract feature space. The loss gradient (red dotted line) is propagated back through the abstraction module and the differentiable renderer.

The OpenDR method is built around the screen-space approximation of the derivative of the rendering process. Gradients due to occlusion effects during this 3D-2D reduction are approximated from the local intensity gradient. In essence, this idea assumes that occluded pixels are similar to their visible neighbors.

In order to simplify gradient computation, we locally linearize the pose:

$$T(\alpha, \beta, \gamma, a, b, c) = T_0 \begin{pmatrix} 1 & -\gamma & \beta & a \\ \gamma & 1 & -\alpha & b \\ -\beta & \alpha & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (6)$$

The rotation part of T is orthonormalized after each optimization step.

The scenes we are interested in feature high levels of occlusion between the individual objects. [6] makes several assumptions in computation of the screen-space gradients. While these assumptions help in simplifying the computation, in real world scenarios, they are often violated. We discuss the assumptions involved, scenarios where these assumptions are violated, and propose solutions for better approximation of the pose gradients.

A. Gradients on Occlusion Boundaries

Occlusion boundaries are highly important for pose refinement, since they offer much information about the scene layout. On the occlusion boundary pixels, OpenDR uses Sobel kernel ($\frac{1}{2}[-1, 0, 1]$) and its transpose to compute the gradient along the horizontal axis and the vertical axis respectively, with the underlying assumption that a shift in the occlusion boundary can be approximated by the replacement of the current pixel by the neighboring pixel (of the other object). However, this assumption is valid only if the occlusion boundary pixel belongs to the object in the foreground. Fig. 4 (a), (b) depict the front view and top view of an example scene where the mug is occluding the can. Translating the mug results in covering or uncovering of can pixels, which is well approximated using the local Sobel gradient. Conversely, translating the can in the background does not result in covering/uncovering mug pixels, rather, more can pixels will become visible or become covered. Thus using the Sobel derivative is incorrect in this case. To address this issue, we detect such cases using the Z-buffer during rendering suppress the Sobel gradient on these pixels. We note that the occluded pixel belongs to the same object in this case, so that zero gradient should be a good approximation.

B. Propagating Image-space Gradients to Object Coordinates

While propagating the image-space gradients to the object coordinates, only gradient from the pixels belonging to the object needs to be propagated. The naïve way to do this is to mask the image-space gradient with rendered object mask. However, in certain situations this means we are ignoring exactly the pixels where a pixel-wise loss function generates high gradients, namely just outside of the rendered object boundary. Figure 4(c-d) illustrates this point.

To address this issue, we propose a dilation of the rendered object mask by one pixel, in order to include gradient information directly outside of the object boundary.

C. Implementation

LightDR uses OpenGL via the Magnum engine[1] for forward rendering. The gradient backpropagation is built on top of PyTorch to facilitate faster computations on GPU. LightDR needs a few milliseconds for forward rendering and around 50 ms for backpropagation of gradients to object poses.

V. Pose Refinement

Armed with the abstraction module and our differentiable renderer, we can tackle the 6D pose refinement problem in cluttered real-world scenes. We first experimented with the architecture depicted in Fig. 5. The 3D scene with the objects in the current estimated pose P is rendered to generate image I_R. The abstraction module (see Section III) is used to generate abstract representations A_R and A_O from the images I_R and I_O. The loss L is computed as the pixel-wise loss between A_R and A_O. Finally, we can derive the gradient of L with respect to the poses P:

$$\frac{\partial L}{\partial P} = \frac{\partial I_R}{\partial P} \cdot \frac{\partial A_R}{\partial I_R} \cdot \frac{\partial L}{\partial A_R} \quad (7)$$

As depicted in Fig. 5, $\frac{\partial I_R}{\partial P}$ is approximated by the differentiable renderer and $\frac{\partial A_R}{\partial I_R} \cdot \frac{\partial L}{\partial A_R}$ is computed by standard backpropagation. During experiments, we noticed that the

[1]https://magnum.graphics/

latter gradient is rather sparse and focuses most of its magnitude on few spatial locations in the image (see Fig. 6). This effect is well-known, for example in the field of adversarial example generation for CNNs [24], [25]. Here, it is highly undesirable, since the differential rendering process works best with smooth, uniform gradients.

To mitigate this issue, we investigated a second pipeline shown in Fig. 1. We use the method described in Section III-C to create meshes with fused surface descriptors. Rendering these meshes directly results in the abstract rendered A_R. In this case, $\frac{\partial L}{\partial P}$ is simpler:

$$\frac{\partial L}{\partial P} = \frac{\partial L}{\partial A_R} \cdot \frac{\partial A_R}{\partial P}. \tag{8}$$

Here, $\frac{\partial A_R}{\partial P}$ is approximated directly by the differentiable renderer. An additional benefit of this variant is that only one forward pass of the abstraction module for A_O is required. This directly translates to significant reduction in the optimization process runtime.

We optimize the object poses with the AdaGrad optimization scheme. This avoids manual tuning of learning rates for translation and rotation parameters—which differ largely in scale. In our experiments, we use a learning rate $\lambda = 1e-2$ with a decay of 0.99. The optimization runs for 50 iterations, which corresponds to roughly $2\,\mathrm{s}$ per frame.

VI. Experiments

We perform our experiments on the YCB Video dataset [5], which consists of 133,936 images extracted from 92 videos, showing 21 different objects in cluttered arrangements. Importantly, the dataset comes with high-quality meshes, which are also used for synthetic data generation by most of the pose estimation methods applied to the dataset [5], [26].

We first qualitatively show the result of the learned mapping into the abstract feature space in Fig. 7. To be able to control primary and secondary effects separately, we rely on our rendering pipeline and generate multiple scenes with random secondary parameters as detailed in Section III-A. Our trained model is able to effectively suppress the background pixels and produces robust, consistent output under changing lighting conditions and camera model parameters.

For a quantitative analysis, we measure the the ADD and ADD-S metrics as in [5] for each object occurrence, which measure average point-wise distances between transformed objects and the ground truth, for non-symmetric and symmetric objects, respectively:

$$\text{ADD} = \frac{1}{m} \sum_{x \in \mathbb{M}} ||(Rx + T) - (\tilde{R}x + \tilde{T})||, \tag{9}$$

$$\text{ADD-S} = \frac{1}{m} \sum_{x_1 \in \mathbb{M}} \min_{x_2 \in \mathbb{M}} ||(Rx_1 + T) - (\tilde{R}x_2 + \tilde{T})||, \tag{10}$$

where R and T are the ground truth rotation and translation, $\tilde{R}$ and $\tilde{T}$ denote the estimated pose, and $\mathbb{M}$ is the set of model points as included in the YCB Video dataset. We aggregate all results and measure the area under the threshold-accuracy curve for distance thresholds from zero to $0.1\,\mathrm{m}$, which is the same procedure as in [5].

We demonstrate pose refinement from the initialization of PoseCNN [5] and the newer method by Oberweger *et al.* [26]. Figure 8 displays qualitative refinement examples, while Table I gives quantitative results. In our experiments, we assume that objects were correctly detected so that we can focus on the problem of refining poses rather than correcting detections. Our pipeline gives consistent improvements across nearly all objects of the dataset for the PoseCNN initialization. Note that we do not compare against the PoseCNN variant with ICP post-refinement, since our pipeline works with RGB only and ICP requires depth measurement. The improvement is especially significant for large and textured objects. On the initializations of Oberweger *et al.* [26], which are already of very high quality, our gains are smaller. We hypothesize that our approach is currently limited by the spatial resolution of the computed feature representation.

Finally, compared to DeepIM [16], our method almost reaches the same overall performance. We note that the experiments performed in [16] apparently started from a better PoseCNN initialization than what was available to us, though the difference seems small. Interestingly, our method obtains significantly better results on a few object classes—suggesting that a combination of the techniques (e.g. by making the abstract representation and computed pose updates accessible to the DeepIM network) could yield further improvements.

To quantify the robustness of our render-and-compare pipeline to the quality of the initialization, we analyzed the basin of attraction of the refinement process. We experimented with 295 scenes from the validation set of YCB-Video dataset ($\sim$10 % of the total validation scenes) by randomly perturbing the translation and rotation components of the ground truth poses to varying degrees and optimizing the perturbed poses. The translation perturbations were uniformly sampled in a range of ± 5 centimeters. Since the impact the translation perturbations has for an object depends of the size of the object, we compute the percentage of pixel overlap between the observed image and the rendered image for an object.

Similarly, we uniformly sample an axis of rotation and an rotation angle in the range ± 45 degrees. The AUC of the

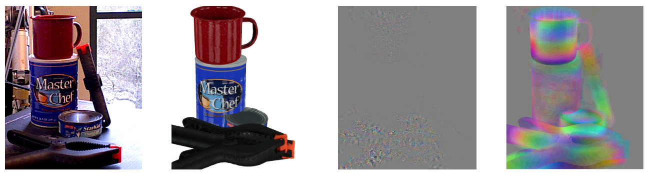

Fig. 6. Gradient of pixel-wise loss w.r.t. the rendered scene in the two pipeline variants. (a) Observed scene. (b) Rendered scene. (c) Render-Abstraction pipeline (see Fig. 5). (d) Abstraction-Render pipeline (see Fig. 1). The actual gradient magnitudes are scaled for better visualization. Gray corresponds to zero gradient.

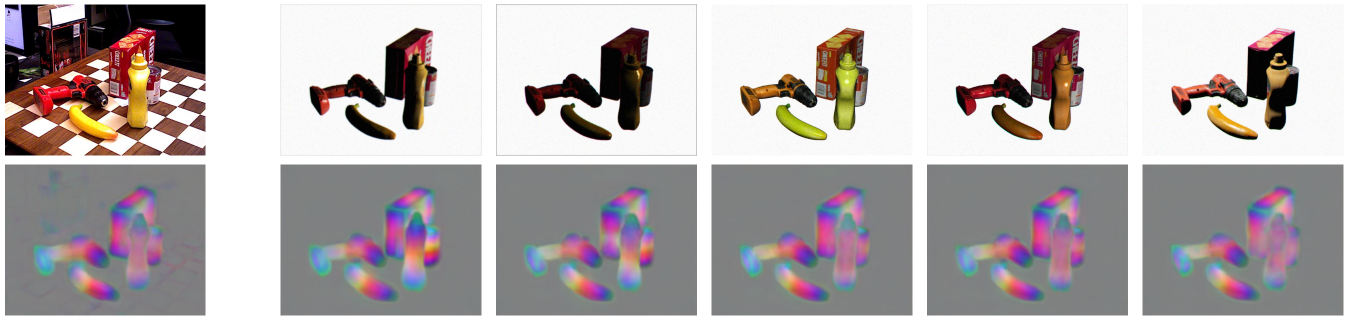

Fig. 7. Learned scene abstraction, removing secondary effects such as background, lighting, shadows, and camera noise. Top row: RGB input scene, bottom row: RGB visualization of the output feature descriptors. The leftmost column shows a real test scene from the YCB dataset, the other columns contain rendered scenes with random lighting and camera noise.

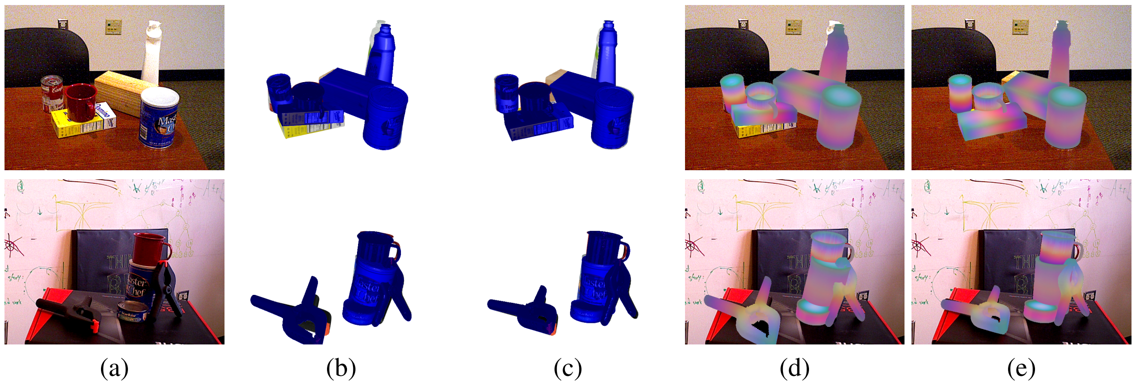

Fig. 8. Qualitative examples from the YCB Video Dataset. (a): Observed scene. (b) and (c): Renderings (blue) with initial and optimized pose parameters, respectively. (d) and (e): Renderings of the feature-annotated meshes in initial and optimized pose, respectively.

TABLE I

POSE REFINEMENT RESULTS ON THE YCB VIDEO DATASET

	PoseCNN [5]		PoseCNN refined (ours)		HeatMaps [26]		HeatMaps refined (ours)		DeepIM [16]	
Object	ADD	ADD-S	ADD(Δ)	ADD-S(Δ)	ADD	ADD-S	ADD(Δ)	ADD-S(Δ)	ADD	ADD-S
master_chef_can	50.2	83.9	63.3(+13.1)	91.7(+7.8)	81.9	91.4	76.7(−5.1)	90.2(−1.2)	65.2	87.8
cracker_box	53.1	76.9	65.3(+12.2)	81.7(+4.9)	83.6	90.0	82.9(−0.7)	89.4(−0.6)	82.6	89.8
sugar_box	68.4	84.2	85.3(+16.9)	92.0(+7.8)	82.1	89.8	86.4(+4.3)	92.2(+2.4)	89.7	93.8
tomato_soup_can	66.2	81.0	59.4(−6.8)	79.9(−1.1)	79.8	89.5	57.4(−22.4)	78.2(−11.3)	81.4	90.1
mustard_bottle	81.0	90.4	86.5(+5.5)	92.3(+1.9)	91.5	95.0	86.7(−4.7)	92.6(−2.4)	90.3	94.4
tuna_fish_can	70.7	88.0	81.1(+10.4)	94.3(+6.3)	48.7	71.7	69.7(+21.0)	85.7(+14.0)	85.4	94.5
pudding_box	62.7	79.1	71.1(+8.4)	83.1(+4.1)	90.2	94.1	68.8(−21.4)	80.7(−13.4)	84.9	91.8
gelatin_box	75.2	87.2	81.5(+6.3)	89.1(+1.9)	93.7	95.9	73.0(−20.7)	82.8(−13.1)	87.7	91.6
potted_meat_can	59.5	78.5	63.7(+4.2)	80.3(+1.8)	79.1	90.0	74.6(−4.5)	87.6(−2.4)	70.0	78.2
banana	72.3	86.0	82.1(+9.8)	91.8(+5.8)	51.7	67.8	68.8(+17.1)	81.0(+13.2)	83.3	92.0
pitcher_base	53.3	77.0	85.1(+31.8)	92.7(+15.7)	69.4	85.0	83.8(+14.4)	92.1(+7.1)	88.7	93.7
bleach_cleanser	50.3	71.6	65.0(+14.7)	80.4(+8.9)	76.2	85.5	78.3(+2.0)	87.6(+2.2)	75.9	86.8
bowl	3.3	69.6	6.5(+3.1)	75.5(+5.9)	3.6	78.1	1.5(−2.1)	66.4(−11.6)	41.5	77.8
mug	58.5	78.2	65.9(+7.4)	84.0(+5.9)	53.9	75.8	57.9(+4.0)	78.9(+3.1)	70.3	86.1
power_drill	55.3	72.7	73.7(+18.4)	85.9(+13.2)	82.9	90.8	81.5(−1.3)	90.4(−0.4)	90.7	94.6
wood_block	26.6	64.3	45.5(+18.9)	73.3(+9.0)	0.0	57.0	0.0(+0.0)	60.3(+3.3)	26.2	60.1
scissors	35.8	56.9	40.0(+4.1)	58.6(+1.7)	65.3	79.6	75.4(+10.1)	85.4(+5.8)	45.5	61.8
large_marker	58.3	71.7	63.9(+5.6)	77.3(+5.6)	56.5	70.2	59.8(+3.3)	70.2(+0.0)	68.1	77.5
large_clamp	24.6	50.2	37.0(+12.4)	65.1(+15.0)	57.2	73.1	75.3(+18.1)	85.6(+12.5)	45.5	72.1
extra_large_clamp	16.1	44.1	25.4(+9.3)	63.7(+19.6)	23.6	54.6	20.4(−3.1)	58.3(+3.7)	29.1	70.0
foam_brick	40.2	88.0	43.3(+3.1)	90.8(+2.8)	32.1	88.9	37.0(+5.0)	92.1(+3.2)	70.5	83.0
ALL	53.7	75.8	62.8(+9.1)	82.4(+6.6)	66.2	82.4	67.0(+0.7)	83.5(+1.1)	70.1	84.2

We report the area under the accuracy curve (AUC) for varying error thresholds on the ADD and ADD-S metrics.

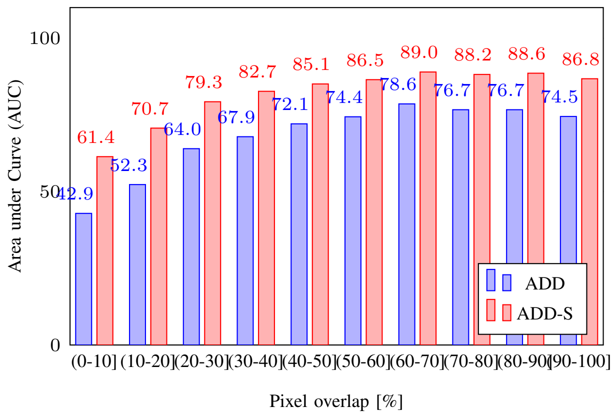

Fig. 9. Basin of attraction in translation dimensions. We show resulting ADD/ADD-S metrics for varying initial 2D overlap of ground truth pose and initial estimate.

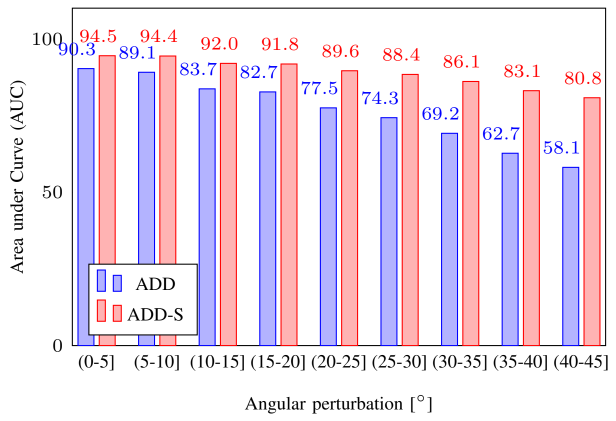

Fig. 10. Basin of attraction in rotation dimensions. We show resulting ADD/ADD-S metrics for varying initial angular perturpations from the ground truth pose.

optimized pose with respect to different overlaps is shown in Fig. 9 and the rotation angle is shown in Fig. 10. Our method is able to robustly handle translation perturpations with almost no loss in accuracy down to 30% remaining overlap. In the rotation experiment, the ADD-S metric is almost unaffected by rotations of up to 45°. The ADD metric drops off more steeply—this is caused by the entirely symmetric objects, where the system has no chance of correcting the perturbation around the symmetry axis.

VII. Conclusion

We introduced a technique for scene abstraction by dense object features learned in a self-supervised way and scene analysis by render-and-compare utilizing a fast differential renderer implementation. Our proposed method yields good results on the challenging YCB Video dataset, where it robustly refines rough initial pose estimates to precise localizations. We further demonstrated its large basin of attraction from perturbed initializations. We see our result as a proof-of-concept for differential rendering in the context of scene analysis. In future work, we will increase performance further and investigate other applications, for example non-rigid registration of category-level models. Due to its formulation the method could also be combined with other iterative refinement procedures, contributing its holistic scene understanding.

References

[1] D. Pavlichenko, D. Rodriguez, M. Schwarz, C. Lenz, A. S. Periyasamy, and S. Behnke, "Autonomous dual-arm manipulation of familiar objects," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2018, pp. 1–9.

[2] T. Klamt, D. Rodriguez, M. Schwarz, C. Lenz, D. Pavlichenko, D. Droeschel, and S. Behnke, "Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1–8.

[3] U. Grenander, *Lectures in Pattern Theory-Volume 1: Pattern Synthesis*. 1976.

[4] ——, *Lectures in Pattern Theory: Volume II: Pattern Analysis*. Springer-Verlag, 1978.

[5] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[6] M. M. Loper and M. J. Black, "OpenDR: An approximate differentiable renderer," in *European Conference on Computer Vision (ECCV)*, Springer, 2014, pp. 154–169.

[7] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D mesh renderer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2018, pp. 3907–3916.

[8] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess, "Unsupervised learning of 3D structure from images," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 4996–5004.

[9] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[10] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable monte carlo ray tracing through edge sampling," in *SIGGRAPH Asia 2018 Technical Papers*, ACM, 2018, p. 222.

[11] S. Liu, W. Chen, T. Li, and H. Li, "Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction," *arXiv preprint arXiv:1901.05567*, 2019.

[12] J. Zienkiewicz, A. Davison, and S. Leutenegger, "Real-time height map fusion using differentiable rendering," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 4280–4287.

[13] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt, "MOFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 1274–1283.

[14] A. Kundu, Y. Li, and J. M. Rehg, "3D-RCNN: Instance-level 3D object reconstruction via render-and-compare," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3559–3568.

[15] P. Moreno, C. K. Williams, C. Nash, and P. Kohli, "Overcoming occlusion with inverse graphics," in *European Conference on Computer Vision (ECCV)*, Springer, 2016, pp. 170–185.

[16] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep iterative matching for 6D pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[17] H. Barrow, J Tenenbaum, A Hanson, and E Riseman, "Recovering intrinsic scene characteristics," *The Journal of Computer and System Sciences*, vol. 2, no. 3-26, p. 2, 1978.

[18] M. F. Tappen, W. T. Freeman, and E. H. Adelson, "Recovering intrinsic images from a single image," in *Advances in neural information processing systems (NeurIPS)*, 2003, pp. 1367–1374.

[19] G. D. Finlayson, M. S. Drew, and C. Lu, "Intrinsic images by entropy minimization," in *European conference on computer vision (ECCV)*, Springer, 2004, pp. 582–595.

[20] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 2, pp. 420–427, 2017.

[21] V. Nekrasov, C. Shen, and I. Reid, "Light-weight refinenet for real-time semantic segmentation," *arXiv preprint arXiv:1810.03272*, 2018.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[23] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense Object Nets: Learning dense visual object descriptors by and for robotic manipulation," *arXiv preprint arXiv:1806.08756*, 2018.

[24] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[25] S. Palacio, J. Folz, J. Hees, F. Raue, D. Borth, and A. Dengel, "What do deep networks like to see?" In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3108–3117.

[26] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3D object pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 119–134.