A Shared Pose Regression Network for Pose Estimation of Objects from RGB Images

Bengtson, Stefan Hein; Åström, Hampus; Moeslund, Thomas B.; Topp, Elin A.; Krueger, Volker

Published in:
2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)

DOI (link to publication from Publisher):
10.1109/SITIS57111.2022.00022

Publication date:
2023

Document Version
Accepted author manuscript, peer reviewed version

A Shared Pose Regression Network for Pose Estimation of Objects from RGB Images

1st Stefan Hein Bengtson[1]
3rd Thomas B. Moeslund
Department of Architecture, Design, and Media Technology, Aalborg University
Visual Analysis and Perception (VAP) Laboratory
Pioneer Centre for AI, Denmark
Aalborg, Denmark
[1]shbe@create.aau.dk

2nd Hampus Åström[2]
4th Elin A. Topp
5th Volker Krueger
Department of Computer Science, Lund University
Robotics and Semantic Systems
Lund, Sweden
[2]hampus.astrom@cs.lth.se

***Abstract*—In this paper we propose a shared regression network to jointly estimate the pose of multiple objects, replacing multiple object-specific solutions. We demonstrate that this shared network can outperform other similar approaches that rely on multiple object-specific models by evaluating it on the T-LESS dataset using the VSD (Visible Surface Discrepancy). Our approach offers a less complex solution, with fewer parameters, lower memory consumption and less training required. Furthermore, it inherently handles symmetric objects by using a depth-based loss during training and can predict in real-time. Finally, we show how our proposed pipeline can be used for fine-tuning a feature extractor jointly on all objects while training the shared pose regression network. This fine-tuning process improves the pose estimation performance.**

***Index Terms*—pose estimation, symmetry, CAD model, differentiable rendering, robotics**

I. Introduction

6D pose estimation entails identifying both the orientation and position of an object. These two pieces of information are useful in multiple scenarios, for instance, the pose of an object could be used for a robotic manipulator to infer possible ways to interact with that object such as grasping.

One aspect that complicates the process of pose estimation is the presence of symmetries in some objects, making it hard or impossible to distinguish some poses from each other. This is exemplified in the T-LESS dataset [1], which features multiple highly symmetric objects, as shown in Fig. 1. The problem of symmetries is further complicated for the type of objects found in T-LESS, as they lack textures that could help resolve such ambiguities.

Another important aspect of pose estimation is the context in which it is used, as there is often a trade-off between the accuracy of the estimates and the inference time. For instance, an approach [2] can produce highly accurate pose estimates but require several seconds to process a single image crop. This will not be ideal for some robotic applications, like bin-picking, where speed is key. Spending several seconds per object in a scene such as the ones shown in Fig. 1, would for many applications be unacceptable. In such a scenario it may be more preferable to use an approach capable of running in real-time at the cost of a lower pose estimation accuracy [3].

Fig. 1: Four scenes from the T-LESS dataset [1] containing highly symmetric objects without any texture. The dataset contains a total of 30 objects across 20 scenes.

This paper concerns the cases where speed is of the essence. Currently, one of the fastest [3], [4] approaches rely on a codebook-based approach [5] which can achieve a high frame rate in most scenarios, making it applicable for purposes requiring real-time execution. This approach has been improved by replacing the codebooks by small object-specific pose regression networks [6], while relying on a pre-trained feature extractor from the previous approach. By using these small regression networks, pose estimation performance is increased and memory consumption is reduced, while maintaining a low inference time. Another benefit of these approaches is that they implicitly account for any symmetries that an object might have by considering the visual similarity of the object when comparing poses.

In this paper we show how a single shared pose regression network can replace multiple object-specific networks [6]. While doing so, we not only improve pose estimation performance, but also reduce complexity, memory consumption and training time.

The main contributions of this paper are:

- We propose a single shared pose regression network, replacing multiple separate object-specific networks. This reduces training time and memory consumption.
- We evaluate our method against two other approaches, multiple object-specific networks and multiple object-specific codebooks.
- We show that fine-tuning the pre-trained feature extractor improves the shared networks pose estimation performance noticeably.
- We show that a shared network can reduce training data needed by a third and memory usage by half, while running in real time and slightly improving VSD recall.

II. Related Work

Traditional methods relying on handcrafted descriptors have been successful in the area of pose estimation in the recent decade, with many of these methods relying on depth information for calculating hand-crafted features based on 3D points [7], [8]. These extracted features are used in various matching and voting schemes in order to produce pose estimates to align the input data with predefined models of the objects. However, computing these features and the matching process are often quite computationally heavy, and it can take several seconds to process a single object.

More recent approaches for pose estimation harness the power of deep learning to produce pose estimates directly from RGB images of the objects of interest [9], [10]. The drawback of these approaches is the huge amount of training required, which prompts the need for massive amounts of training data and expensive computing power.

Furthermore, the process of training these approaches may also prove troublesome for objects with symmetries due to the resulting pose ambiguities. For instance, training samples may be labeled as having widely different poses despite looking visually very similar due to symmetries, such as in Fig. 2a and 2b. Some approaches try to counteract this issue by relying on pre-defined symmetries for each object, such that the pose with lowest error is always chosen from the set of symmetric poses [9], [10]. However, some symmetries are not easily pre-defined as they might occur due to occlusion as illustrated in Fig. 2c. Furthermore, identifying pre-defined symmetries often relies on a manually selected threshold specifying whether two poses are similar enough to be considered symmetric [4], [9]. This approach does hence fail to encompass the strength of the symmetries, for instance, how visually similar two poses might be. For instance, the poses in Fig. 2a and 2b may or may not be considered a symmetry depending on how this threshold is set.

Yet another solution is to avoid the issues of symmetries by confining the set of poses in the training data such that any symmetries are avoided altogether [11], [12]. This approach is similar to relying on pre-defined symmetries, as it also relies on specific knowledge about the symmetries of each object beforehand and therefore will fail to encompass symmetries caused by occlusion.

(a) Rotation: 0^o. (b) Rotation: 180^o. (c) Occluded.

Fig. 2: Examples of the same object in multiple poses. The poses depicted in a) and b) appear visually similar despite a difference of 180^o due to the semi-symmetric nature of the object. Finally, the object in c) is occluded causing the remaining visible part to have a high visual similarity with the poses depicted in both a) and b).

A lot of these problems related to symmetric objects can be avoided by focusing on the visual similarity of the object in the different poses instead of focusing on the actual poses. Examples of such includes using a codebook-based approach to identify visually similar poses [5] or training a model to predict poses which just has to be visually similar to the ground truth pose [6]. Both of these approaches can implicitly handle symmetric objects and do hence not require pre-defined symmetries for each object.

Recently, pose estimation methods have started to rely on deep learning for solely extracting local feature descriptors across an RGB image of an object. These local descriptors are then used to estimate the pose of the object by solving the PnP (Perspective-n-Point) problem in a RANSAC-like fashion [2], [3], [13]. The actual process of estimating the pose is hence not part of the training of the DNN (deep neural network) thereby avoiding the issue of pose ambiguities caused by object symmetries while achieving state-of-the-art pose estimation performance [2]. However, the process of estimating the pose from these local descriptors is often a slow process which could take several seconds per object, just like some of the traditional approaches based on hand-crafted 3D features [7], [8]. These approaches relying on local features are hence either impractical or infeasible for real-time applications.

Approaches with a decent pose estimation performance but with low inference time are hence still needed [3]. This is especially important in scenarios where real-time execution is important, which is often the case for e.g., robotic applications. Using a codebook-based approach [5] can easily achieve a frame rate of over 20 FPS in most scenarios, making it applicable for purposes requiring real-time execution. This approach was subsequently further improved by replacing the codebooks by small object-specific pose regression networks [6] which preserved the low inference time while reducing memory usage and increasing the pose estimation performance as well.

III. Method

The work in this paper is based on an approach relying on multiple object-specific pose regression networks [6] and

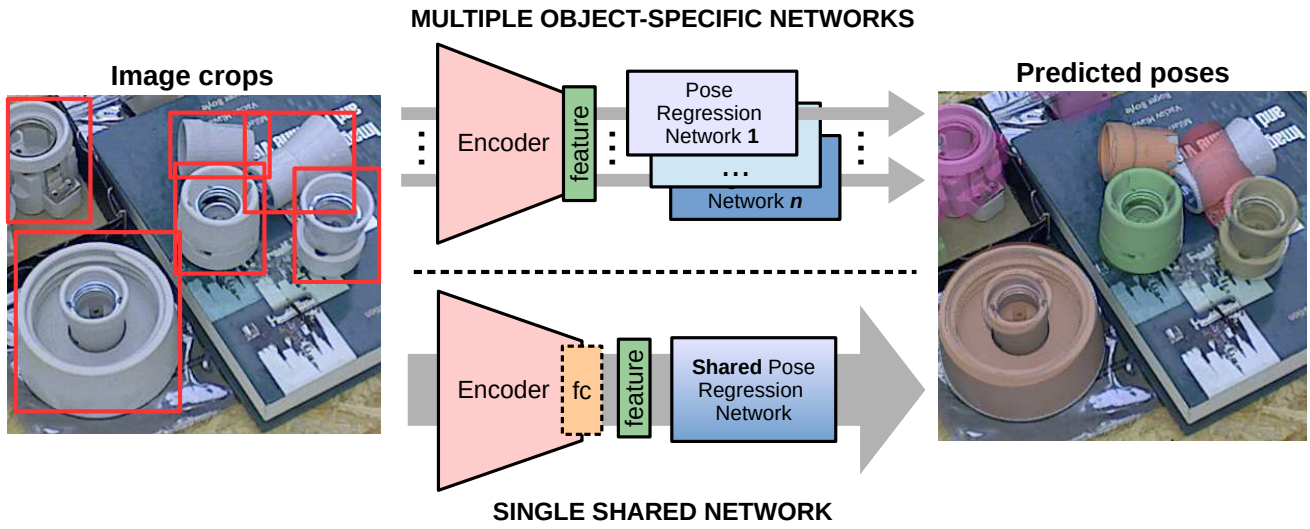


Fig. 3: *Top*: Multiple object-specific pose regression networks are trained independently of each other [6]. These networks rely on a shared feature extractor in the form of the pre-trained encoder [5]. *Bottom:* We propose to replace all these object-specific networks with a single shared pose regression network. Furthermore, we suggest fine-tuning the last fully-connected layer of the pre-trained encoder while training the shared network jointly on all objects.

shows how this approach can be improved and simplified by replacing the many object-specific networks with one shared pose regression network, as shown in Fig. 3. We show that it is possible to train this single shared pose regression network for all the objects while simultaneously increasing the overall performance. Both approaches are based on the same feature extractor, in the form of the pre-trained encoder from a codebook-based approach [5]. This paper also includes an exploration of the benefits of fine-tuning parts of the pre-trained encoder for the pose regression task, while training the shared network.

The proposed method assumes input from an object detector in terms of a bounding box and an associated object ID. This is similar to the codebook-based approach [5] and the approach using multiple object-specific networks [6].

A. Network Architecture

The structure of the proposed shared pose regression network is illustrated in Fig. 4. The first part of the network resembles the object-specific networks [6], where a pre-trained encoder from the codebook-based approach [5] is used as a feature extractor for the network. The input for the pose regression part of the network is thus the latent space produced by this encoder. The size of this feature vector, or latent space, is set at 128 as this is considered optimal in previous work [5].

We propose to fine-tune the last fully-connected layer of the encoder based on the hypothesis that it could provide a better feature representation for doing pose estimation later on. This fine-tuning of the encoder is done while training the shared pose regression network.

During training, synthetic RGB images of various objects in random poses are continuously rendered from CAD models of the objects and used as training data [5]. This avoids the issue of having to provide massive amounts of hand-labeled data while training the network, and have proven to generalize well to real data [5], [6].

Each of the estimated poses are represented using a 6D vector, as previous studies have shown that it is more stable for pose regression tasks than other representations such as quaternions and rotation matrices [14].

Finally, similar to the approach relying on multiple object-specific pose regression networks [6], only the orientation of the object is estimated by the network. Estimating the translation of the object is not included in the proposed pipeline but can be done similarly to how it was solved in connection to the codebook approach [5], or included as a task for the shared network in future work.

The proposed shared network is able to predict poses for multiple objects by expanding the size of the final output layer, as illustrated in Fig. 4. The network therefore always outputs a multi-pose [6] estimate for each of the k different objects, where k is the number of objects the network is trained to estimate poses for. Some of the weights in the final fully-connected layer are therefor tied to a certain object and are class specific, while the rest of the model is shared between all the objects.

The size of the output layer in the proposed shared network is scaled in a linear fashion with k. One drawback of this approach is that the number of parameters in the model will increase as the number of object categories increases. However, this increase is negligible compared to the rest of the pipeline. This can be seen in Section IV-B, where the overall memory usage of the proposed approach is compared against having multiple object-specific networks instead.

Training the proposed shared network relies on a masking

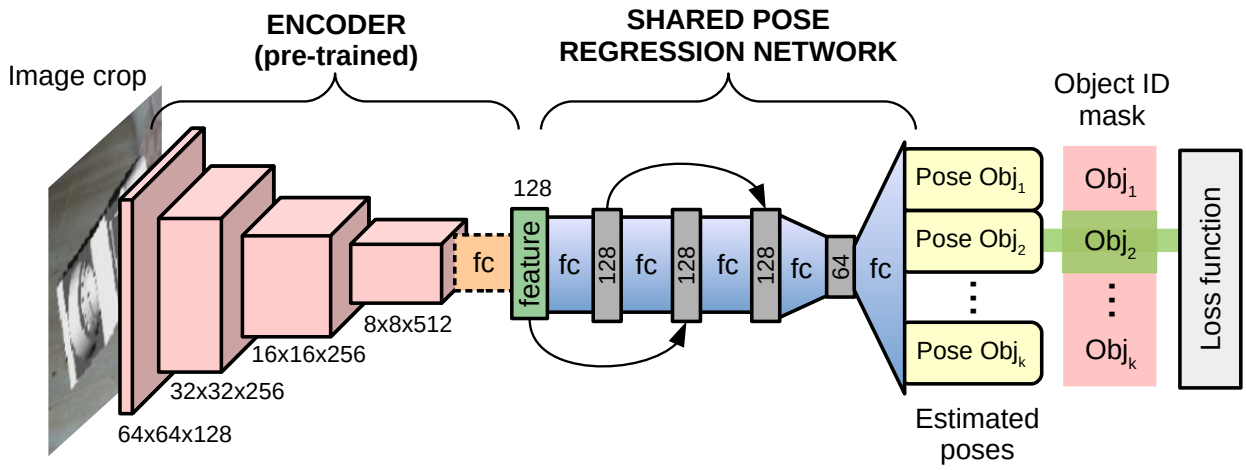


Fig. 4: The architecture of the shared posed regression network, including the pre-trained encoder used as feature extractor [5]. The network is nearly identical to the original pipeline [6] and includes several skip connections as these were found to increase performance. The exception is the final layer of the network which has been modified to output multi-pose estimates for all the k different object categories, regardless of the class ID of the input. Additionally, a masking scheme is introduced to ensure that only the estimated poses for the correct object ID is propagated further in the pipeline. It is assumed that the object ID is available from a prior detection step.

scheme to ensure that only the predicted poses for the correct objects are propagated to the loss function. This is illustrated in Fig. 4, where only one of the pose estimates propagates further to the loss function after applying the object ID mask. This approach assumes that an object ID is provided from a prior object detection step, both during training and inference, in order to apply the mask correctly.

The actual loss is calculated by comparing depth maps of the object in different poses in the same way as the individual pose regression approach [6], as illustrated in Fig. 5. This depth-based loss is heavily inspired by the VSD (Visible Surface Discrepancy) metric [15], [16], that is commonly used in pose estimation benchmarks. The idea is to render depth maps of the different objects in the estimated poses and in the groundtruth poses, respectively. These depth maps are then compared in a pixel-wise approach to identify the visual similarity of the estimated versus ground truth poses. The main motivation of this approach is that it implicitly accounts for any symmetries that the objects might have as the depth maps will look similar in that case and hence incur a small error. These depth maps are produced using differential rendering [17] in order to allow back-propagation when training the shared pose regression network.

Furthermore, each estimation consists of multiple hypotheses in order to avoid getting stuck in local minima, which can easily occur in this depth-based loss [6]. Each estimate includes n hypotheses for each of the estimated poses along with a predicted confidence for each hypothesis $w_1, ..., w_n$. This is illustrated in Fig. 5, where $n = 3$. The final loss is essentially then a weighted average amongst the different pose hypotheses using the predicted confidences as the weights. The parameter, γ, ensures that each pose hypothesis always contributes a little to the loss, so that back-propagation updates all hypotheses during training, and the regularization term, L_{pose}, ensures that the multiple pose hypotheses do not become too similar, which would defeat the purpose of the multiple hypotheses [6]. Finally, this depth-based loss is only used during training of the shared network. No depth maps are rendered during inference.

IV. Evaluation

Our approach is evaluated on the T-LESS dataset [1], containing 30 objects that are characterized by a lack of texture and by being highly symmetric. This combination, no texture and many symmetries, is what makes this dataset challenging.

The network is trained on synthetic data using the same scheme as in the original pipeline using multiple object-specific networks [6] but with a fixed learning rate of $= 0.001$ instead of an adaptive one [18]. A fixed learning rate is used as it allows training the network for an unknown amount of epochs until it converges, since the adaptive learning rate scheme originally used for training the object-specific networks requires a fixed number of epochs that has to be specified prior to training. The network is trained until convergence, with 100,000 newly generated synthetic samples each epoch. These samples are generated randomly from the 30 objects in the T-LESS dataset using the CAD models supplied in the dataset.

The vertices of the CAD models are normalized so that all objects have the same length along their respective largest dimension. This normalization was found necessary in order to avoid some objects dominating the training phase by being bigger than other objects and therefore more likely to incur a bigger loss. Such considerations, regarding some objects being more dominant than others, is not a concern when training a separate network for each object. All the other parameters used

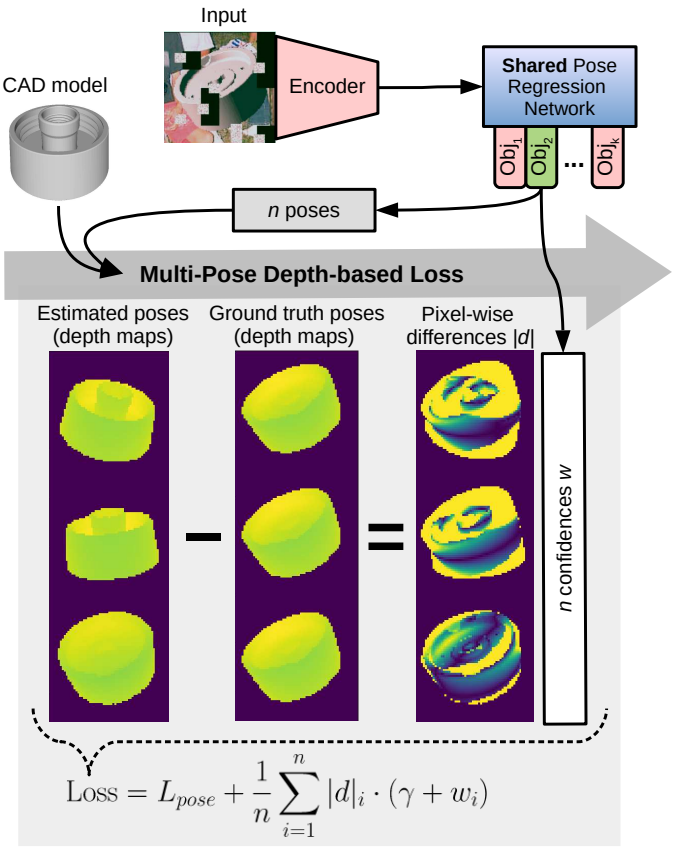
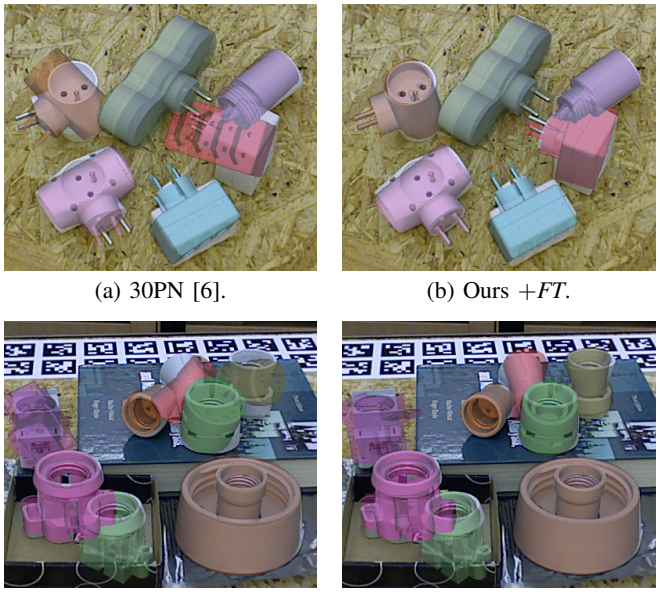


Fig. 5: The depth-based loss [6] used when training the shared pose regression network. It avoids any symmetry-related issues by using rendered depth maps to measure the similarity of poses. Multiple possible hypotheses are predicted for each pose estimate to avoid getting stuck in local minima and the final loss is a weighted average of these hypotheses using their confidences as weights. The parameter, γ, ensure that each hypothesis contributes to the loss so that it gets updated by back-propagation. The regularization term, L_{pose}, is included to incur a high loss if the pose hypotheses becomes to similar.



(a) 30PN [6]. (b) Ours $+FT$. (c) 30PN [6]. (d) Ours $+FT$.

Fig. 6: Examples of pose estimates from the proposed shared network and an approach using object-specific networks [6]. Using the different pose estimates, CAD models are plotted on top of the images from the T-LESS test dataset. The colorization is solely for illustration purposes.

during training are identical to the ones used for training the many object-specific pose regression networks [6].

The shared pose regression network described above is evaluated in two variations; with and without fine-tuning of the last layer in the encoder that provides the input to the shared network. We hypothesize that fine-tuning the last layer of the encoder used for feature extraction can increase the performance of the shared pose regression network. Jointly fine-tuning the encoder on all objects is easily done as only one single shared network is trained.

As a comparison, we note that when training multiple object-specific networks [6] it is complicated to jointly fine-tune the encoder in a similar way. It may be possible to train all the object-specific networks concurrently, in order to jointly fine-tune the same encoder in the process. However, such an approach is complex and requires hardware capable of training the many networks concurrently; 30 networks in case of the T-LESS dataset. Alternately, a separate encoder can be fine-tune for each specific object, but this increases the complexity of the system, both in terms of parameters to be trained and in terms of increased memory consumption. However, fine-tuning the encoder in the context of the shared pose regression network proposed in this paper does not incur any of the drawbacks mentioned above and we can therefore evaluate this network both with and without fine-tuning.

A. Results - Pose Estimation

We evaluate our approach on the T-LESS dataset using the VSD metric [15], [16], both with ($+FT$) and without ($\div FT$) fine-tuning of the encoder. Both are trained until convergence, which is achieved at 200 epochs. Our approach is compared against an approach using multiple object-specific pose regression networks [6] and an approach using multiple object-specific codebooks [5], as shown in Table I. Both ground truth translations and object IDs are used during the evaluation, as both the proposed approach and the multiple object-specific networks approach [6] do not include translation estimation nor object classification. This is done for all the evaluated approaches to ensure a fair comparison.

From these results, it is clear that fine-tuning parts of the encoder improves the performance noticeably as our approach outperforms both the codebook-based approach [5] and the object-specific model approach [6] on average, when fine-tuning is included. Furthermore, the proposed approach also results in the best performance for 15 out of the 30 objects found in the T-LESS dataset. Without fine-tuning of the

TABLE I: VSD recall on the T-LESS dataset for our proposed shared pose regression network. The results with (*+FT*) and without (*÷FT*) fine-tuning of the encoder are reported along with previously published results [6] for the object-specific approaches using either 30 codebooks (30CB) [5] or 30 pose regression networks (30PN) [6].

Object	30CB [5]	30PN [6]	Ours *+FT*	Ours *÷FT*
01	37.82	51.84	**54.00**	41.2
02	51.88	**63.74**	62.12	54.88
03	62.87	71.53	**73.03**	60.25
04	56.00	62.66	**67.14**	56.9
05	77.18	**80.82**	76.26	68.47
06	68.04	66.71	**72.27**	55.99
07	65.18	**65.68**	57.16	50.5
08	**63.11**	61.21	55.21	49.1
09	**68.96**	55.66	53.24	51.87
10	**58.55**	54.14	55.79	31.7
11	**52.15**	51.48	48.09	42.23
12	**62.19**	56.58	54.45	47.79
13	63.56	64.21	**69.19**	59.36
14	57.29	63.01	**67.89**	59.57
15	64.91	66.37	**71.98**	56.3
16	75.82	73.16	**78.25**	71.91
17	76.62	**77.72**	76.77	73.79
18	**71.26**	62.71	61.97	53.26
19	51.19	54.15	**57.50**	44.89
20	40.71	35.96	**43.71**	33.4
21	43.25	43.31	**47.63**	35.1
22	**38.15**	32.03	37.62	22.08
23	39.18	**56.68**	55.50	45.58
24	58.97	61.93	**63.64**	56.93
25	**69.86**	63.08	63.71	52.62
26	57.94	58.87	**60.24**	55.22
27	68.09	**77.62**	69.28	74.67
28	68.06	**73.33**	69.23	69.52
29	76.43	80.67	**83.48**	77.68
30	77.81	83.41	**87.42**	82.42
mean	60.77	62.34	**63.13**	54.51

encoder, the shared pose regression network performs worse than the other two approaches.

Examples of pose estimates produced using both the proposed shared network and multiple object-specific networks are shown in Fig. 6. In the first scene (Fig. 6a and 6b) the predicted poses appear similar for most objects. Exceptions are object 20 (red) and object 21 (orange) where the object-specific models fail to produce feasible pose estimates. The shared network, on the other hand, produces reasonable pose estimates in both cases. Examples like these contribute to the discrepancy in performance in Table I, where the shared network performs the best on both these objects.

In the second scene (Fig. 6c and 6d) both approaches appear to perform similarly on the objects in the front region of the scene. Both approaches also struggle with object 10 (magenta, top left) but this particular object is in general difficult, as seen in the results reported in Table I. However, the pose prediction from the two approaches differs for the group of objects in the upper right corner, consisting of object 13 (orange), 14 (red), 15 (yellow) and 16 (green). In this case, the shared network produces pose estimates which are better aligned with the input images, particularly for the occluded objects where the multi-network approach fails.

TABLE II: Summary of the main characteristics of our approach with fine-tuning (*+FT*) compared to using codebooks [5] and multiple pose regression networks [6]. **The number of training samples is not reported for the codebook-based approach as only the encoder requires training and is assumed to come pre-trained for all approaches.*

	Avg. VSD recall	Inference time	Memory usage	Training samples
30CB [5]	60.77	7.0ms	1365MB	NA*
30PN [6]	62.61	**6.2ms**	33MB	60M
Ours *+FT*	**63.13**	6.4ms	**16.6MB**	**20M**

B. Results - Other Metrics

Besides the pose estimation recall improvement, the proposed approach also reduces the complexity of the system by using a single shared model instead of multiple different ones. The number of parameters, the main contributor to memory usage during inference, is reduced. In the case of the 30 objects in the T-LESS dataset, the reduction in memory consumption is $\approx 51\%$ compared to multiple object-specific networks [6] and $\approx 98\%$ compared to using codebooks [5].

Additionally, using a single shared model reduces training time. The proposed approach was trained for 200 epochs with 100k samples each, amounting to 20 million samples in total. For comparison, each object-specific pose regression network was trained for 200 epochs with 10k samples each [6], resulting in 2 million samples per object. Training 30 separate pose regression networks thus requires three times as many samples as the shared pose regression network.

Finally, it takes $\approx 6.4ms$ to estimate the pose of an object during inference, for the proposed shared pose regression network. This is comparable to the inference time for the approach using multiple object-specific networks ($\approx 6.2ms$) and slightly better than the codebook-based approach ($\approx 7.0ms$). Note that all the timings in terms of the inference time exclude object detection, which is a necessary prior step for all three approaches. Finally, all timings are measured using the same hardware (i7-7700k and GTX1060). Thus, it is possible to achieve a frame rate of 20 FPS in terms of the pose estimation for scenes with 7 objects or less, even if all estimations are done in sequence.

Finally, the different characteristics for the evaluated approaches are summarized in Table II. This includes both pose estimation performance in terms of average VSD recall, inference time, memory usage and number of samples required during training, as discussed in detail in previous sections.

V. Conclusion

This paper proposes a shared regression network for pose estimation of different objects, and shows that it can replace approaches with several object-specific solutions. This shared network is evaluated on the T-LESS dataset and a comparison is made to estimators with multiple object-specific models, either in the form of codebooks or pose regression networks. Our approach achieved the highest overall pose estimation recall by fine-tuning the pre-trainer encoder used for feature extraction while training the shared network. This shared network also offers a less complex solution, with fewer parameters and less memory usage, and it requires less training than the method with multiple object-specific networks. We do this while maintaining the main properties of the two other approaches, as our method handles symmetric objects similarly and has a low inference time, making it suitable for real-time applications. These results indicate that our shared model is preferable over approaches relying on multiple object-specific solutions for pose estimation.

VI. Future Work

A way to improve the presented work could be to consider temporal information, based on the main assumption that the pose of an object does not change in a fraction of a second. Either using various filters [19] or by integrating multiple estimates, in the form of multiple view-points [9]. The benefits of these approaches are promising given the low inference time of the proposed approach, making it possible to produce many pose estimates fast.

Another idea for future work is to include translation estimation, which the approach presented in this work currently lacks, just like the approach using multiple networks [6]. One way could be to infer translation for objects from the size of their bounding boxes in relation to the known size of the objects, from, e.g., the CAD models. However, this approach is very sensitive to noise in the bounding boxes and hence the object detector used [5], [20]. This issue could be counteracted by training a model to estimate adjustments to the bounding box of each object [10].

Yet another avenue for further research could be to expand the presented approach to also predict object IDs as it currently relies on a prior object detection step for this information. Estimating object IDs as part of the shared pose regression network could be based on the idea of visual similarity from depth renderings, just like the pose estimation. Doing so may prove beneficial as wrong predictions in terms of the object ID would be punished less harshly if the objects are visually similar and vice versa.

Finally, it would be interesting to further explore the impact of fine-tuning the pre-trainer encoder, as is essential for the performance of the shared pose regression network in this work. Exploring how similar fine-tuning would impact other approaches is thus another obvious path for future work.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP).

References

[1] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *WACV*, 2017.

[2] R. L. Haugaard and A. G. Buch, "Surfemb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings," 2021.

[3] T. Hodan, D. Barath, and J. Matas, "EPOS: Estimating 6d pose of objects with symmetries," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2020.

[4] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, "BOP challenge 2020 on 6d object localization," in *ECCV Workshops*, 2020.

[5] M. Sundermeyer, M. Durner, E. Y. Puang, Z.-C. Marton, N. Vaskevicius, K. O. Arras, and R. Triebel, "Multi-path learning for object pose estimation across domains," in *CVPR*, June 2020.

[6] S. H. Bengtson, H. Astrom, T. B. Moeslund, E. A. Topp, and V. Krueger, "Pose estimation from RGB images of highly symmetric objects using a novel multi-pose loss and differential rendering," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sep. 2021.

[7] B. Drost, M. Ulrich, P. Bergmann, P. Härtinger, and C. Steger, "Introducing mvtec ITODD - A dataset for 3d object recognition in industry," in *2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2200–2208.

[8] J. Vidal, C.-Y. Lin, X. Lladó, and R. Martí, "A method for 6d pose estimation of free-form rigid objects using point pair features on range data," *Sensors*, vol. 18, no. 8, p. 2678, Aug. 2018.

[9] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, *CosyPose: Consistent Multi-view Multi-object 6D Pose Estimation*, 11 2020, pp. 574–591.

[10] Z. Li, G. Wang, and X. Ji, "CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019.

[11] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6d: Making RGB-based 3d detection and 6d pose estimation great again," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017.

[12] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3d object pose estimation," in *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 125–141.

[13] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019.

[14] Y. Zhou, C. Barnes, L. Jingwan, Y. Jimei, and L. Hao, "On the continuity of rotation representations in neural networks," in *CVPR*, June 2019.

[15] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, "BOP: Benchmark for 6D object pose estimation," *ECCV*, 2018.

[16] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6d object pose estimation," in *ECCV Workshops*, G. Hua and H. Jégou, Eds., 2016, pp. 606–619.

[17] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, "Pytorch3d," https://github.com/facebookresearch/pytorch3d, 2020.

[18] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019.

[19] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "PoseRBPF: A rao–blackwellized particle filter for 6-d object pose tracking," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1328–1342, Oct. 2021.

[20] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *ECCV*, September 2018.