3DPoseLite: A Compact 3D Pose Estimation Using Node Embeddings

Meghal Dani
TCS Research, New Delhi
dani.meghal@tcs.com

Karan Narain
TCS Research, New Delhi
karan.narain@tcs.com

Ramya Hebbalaguppe
TCS Research, New Delhi
ramya.hebbalaguppe@tcs.com

Abstract

Efficient pose estimation finds utility in Augmented Reality (AR) and other computer vision applications such as autonomous navigation and robotics, to name a few. A compact and accurate pose estimation methodology is of paramount importance for on-device inference in such applications. Our proposed solution **3DPoseLite***, estimates pose of generic objects by utilizing a compact node embedding representation, unlike computationally expensive multi-view and point-cloud representations. The neural network outputs a 3D pose, taking RGB image and its corresponding graph (obtained by skeletonizing the 3D meshes [31]) as inputs. Our approach utilizes* `node2vec` *framework to learn low-dimensional representations for nodes in a graph by optimizing a neighborhood preserving objective. We achieve a space and time reduction by a factor of* $11\times$ *and* $3\times$ *respectively, with respect to the state-of-the-art approach, PoseFromShape [50], on benchmark Pascal3D dataset [48]. We also test the performance of our model on unseen data using Pix3D dataset.*

1. Introduction

Augmented Reality (AR) enriches user experience by blending virtual elements with the real environment that contains them. Environment-aware AR applications tend to be more intuitive and user-friendly than the applications that aren't, with features such as plane detection and object registration providing the distinction. In the case of guided surgeries, medical practitioners can benefit from accurate 3D model registration. It can allow accurate patient positioning and instrument positioning for precise drug delivery to the target with minimal invasion [35].

While most modern augmented reality toolkits, both open source and commercial, such as Vuforia [5], Metaio (now with Apple) [4], Wikitude [6], AugmentedPro [2], Diotasoft [3] and, ARKit [1] provide support for object pose estimation, their capability and robustness varies and are typically opaque in terms of underlying algorithms and implementation techniques, and scope for customization [34]. For instance, having a custom registration pipeline allows users to control the extent of overlap of the 3D model with the target during the matching procedure in model registration. In 3D image target registration, the main problem to be addressed is the accurate 3D pose estimation.

This paper focuses on and aims to provide a novel technique for efficiently performing object registration, which is the process of estimating the 3-dimensional pose of a physical object to accurately overlay digital content such as annotations, 3D models, and animations for an immersive user experience. Our method utilizes 3D shape information along with the RGB images. Existing literature employs 3D shapes for rigid body objects in the form of meshes, point clouds or multi-view images [50], [45], [40].

To the best of our knowledge, this is the first work that employs graph representation generated in an unsupervised manner from skeletons, to represent 3D shapes for generic rigid body objects, making our framework category-agnostic. We repurpose *node2vec* embedding used mainly in NLP literature [36] and we leverage these node embeddings in order to provide light-weight intermediate representations of 3D models of the object whose pose is to be estimated. We then make use of a combination of custom and standard Convolutional Neural Networks (CNNs) to match this representation with the visual scan data of the object.

Our intuition is that 3D shapes capture the underlying semantic and compositional information of objects, and using them not only facilitates accurate registration but is also useful in making the registration category-agnostic. In a real-world scenario, we need systems that are ready to handle all kinds of objects. But it is difficult on a human level to train our models on all possible object categories that exist. Its the need of the hour to develop methods that don't require all object types during training. Consider a robotic arm that interacts with objects in front of it and places all of them such that the robot can now see them in the same orientation i.e., in the same view. Ideally, it should be able to align all objects seen or unseen into the required orientations. This can be achieved when we strive to make our models such that they don't require re-training on seeing new objects and can make decisions on their own. Introducing

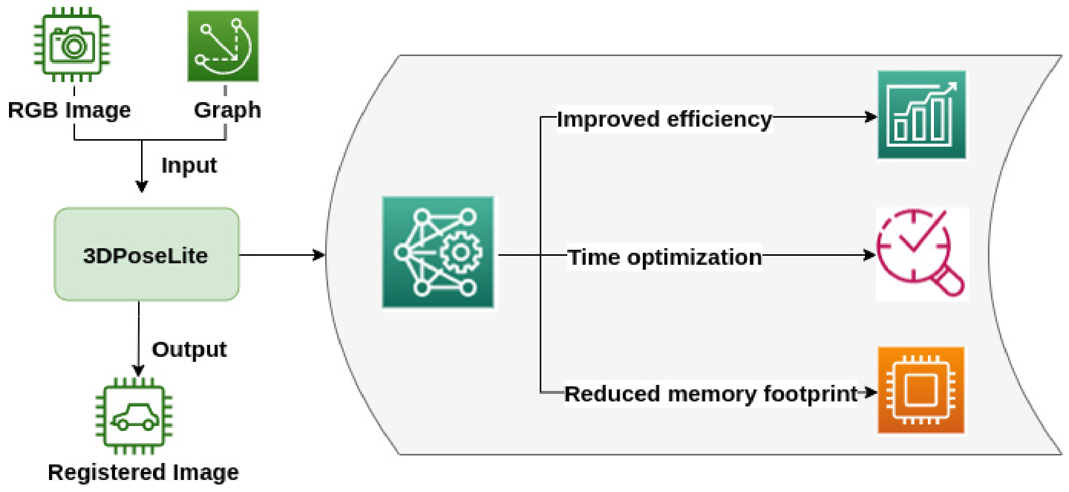

Figure 1: Abstract Diagram: Our proposed framework, 3DPoseLite aims to estimate pose of an object given an RGB image and its associated 3D model's graph representation as input. The task to estimate pose and obtain a registered image as output is accomplished in a lesser time frame with reduced memory footprint, and with similar accuracy as compared to state-of-the-art models.

3D shape information along with RGB data has proved to solve this problem [50]. We claim that the general shape information of an object can be captured with a few nodes and need not be a combination of thousands of points on a point cloud. Moreover, graphs provide concise and easily computable information as compared to point-clouds and multi-view [39] [43]. The invariance to rotation, translation and scaling make graphs an intuitive choice for representing 3D objects. The major contributions of our work are:

1 To the best of our knowledge, our approach, *3DPoseLite*, is the first work in pose estimation that employs unsupervised graph embeddings, `'node2vec'`, to represent 3D shapes for generic rigid body objects.

2 *3DPoseLite* utilizes a compact graph representation that encodes a 3D shape and is able to reduce the memory footprint and time by $11\times$ and $3\times$ respectively, with respect to the state-of-the-art for instance, [50].

2. Literature Survey

Pose Estimation is a fundamental problem in AR and computer vision . Commonly used techniques for the process, extract features from the input 2D-RGB image and match them with the 3D models and finally align these 3D models with the target image. Pose Estimation with 3D to 2D correspondences is a Perspective-n-Point (PnP) algorithm [32]. This algorithm gained a lot of interest among researchers with usage in many problems [25] [49]. However, the method suffers from heavy online run-time computation cost and does not deal with outliers efficiently [17]. Recent studies show that dealing with outliers is an NP-Hard [11] problem. The most common mechanism to deal with outliers is to use PnP with RANdom SAmple Consensus (RANSAC) [18] in loop.

The method is robust with textured objects but fails in case of occluded scenes and almost featureless objects. Another popular algorithm used for registration in varying fields of tasks is iterative closest point (ICP) [8] algorithm.While it has been relatively successful, this method is susceptible to converging to local minima [10], [49] and, therefore, a close initial manual alignment is necessary for convergence. Deformable Part Model (DPM) [16] was also introduced and found to be very efficient. Though these algorithms are reported to work quite well, they do not operate in real-time which is crucial for AR applications.

Recently, CNNs [19], [30] have been shown to outperform Deformable Part Model (DPM) [16] based methods for recognition tasks [21], [12], [28]. While DPMs explicitly model part appearances and their deformations, the CNN architecture allows such relations to be captured implicitly using a hierarchical convolutional structure. Girshick et al. [22] argue that DPMs could also be thought of as a specific instantiation of CNNs and therefore training an end-to-end CNN for the corresponding task should outperform a method which instead explicitly models part appearances and relations. Since then, a multitude of work [29], [40], [45], [20] has been done in development of end-to-end CNN models to learn correspondence of 3D model to 2D image. PoseCNN [49] was one of the initial works in this domain and delivered promising results with images and annotation data. A few researchers made use of segmentation masks instead of annotation data. But these binary masks lacked texture, color and other relevant information which presents difficulties in situations where such information is needed to resolve ambiguities in the mask representation [47]. They relied heavily on segmentation accuracy and did not do well with occluded image datasets.

These approaches were category dependent i.e., they could identify pose only for those objects on which the end-to-end model has been trained on. Y. Xiao et al. [50] for the first time introduced a category agnostic model that made use of 3D shape information along with RGB images to define pose. The results shown were quite promising in AR. The model was unfortunately quite complex and heavy as it required Multi-Views to be generated during training time and slowed down the performance. Our work tries to resolve this issue by proposing an approach for a light model which can be easily used on frugal devices and have improved performance in a memory constrained environment. We represent 3D models in form of graphs using node-embeddings [24] as part of this approach. Graphs have been studied well in humans [46], [53], [52] but not in rigid body objects. Our approach is intended to work on hand-held and head-mounted devices or in particular, where the memory constraints are obvious. Despite the compact representation, our accuracy of pose matches the state-of-the-art technique, *Pose From Shape* [50].

3. 3DPoseLite

We propose *3DPoseLite*, a light-weight pose estimation approach that utilises 3D shapes (in the form of graphs) and images to extract deep features using CNN. Refer Figure 3 that details the architecture. Instead of using raw graphs, we convert them into node embeddings using a semi-supervised algorithm, *node2vec* [24] inspired from word embeddings [37] used in natural language processing literature. The pose is estimated combining the node embeddings with the RGB image using a early fusion strategy. Later sub-sections present details of *3DPoseLite*; we provide additional details on graph representation and node embedding technique, loss functions, and implementation details.

3.1. Advantages of graph representation

High-resolution 3D shape representations such as Point Cloud and Multi-View can be highly expensive both in terms of storage and processing. While these 3D representations like Pointcloud and Multi-View are needed for tasks like high-fidelity rendering and 3D printing, other tasks, such as shape retrieval, only require access to a specific subset of the shape properties. Such tasks are favored by a compact shape representation that encodes the key properties for the tasks at hand in a computationally efficient way [39]. In this regard, a skeleton representation gives a compact and effective shape abstraction. Ideally, skeletons preserve topological information and are stable under small deformations and Euclidean transformations. In fact, the usage of skeleton based graphs to search for similar objects in a database is more feasible than simply comparing point clouds or thousands of triangles in a mesh [43].

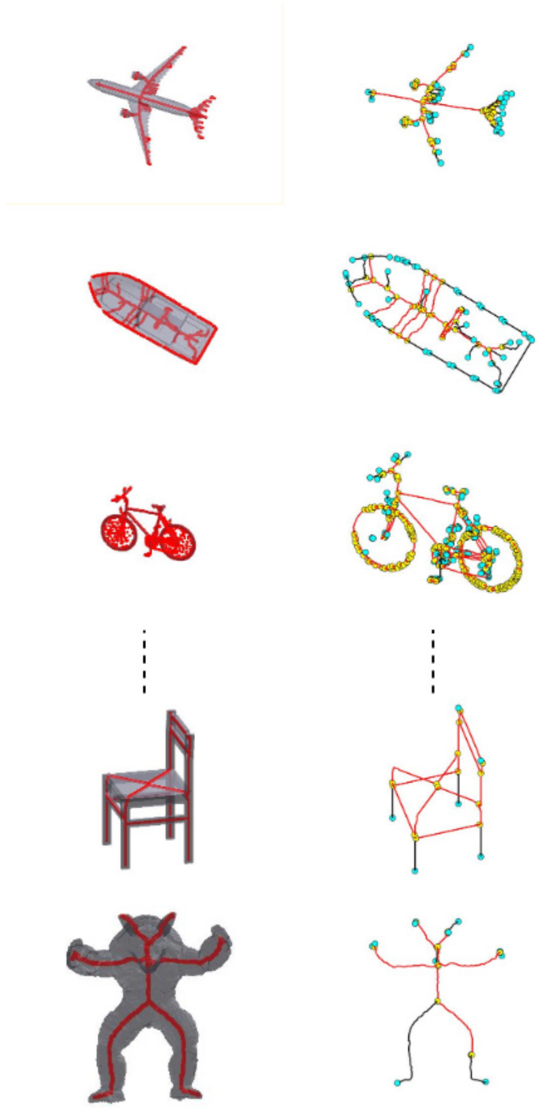
Figure 2: The figure in the left columns provides visual illustration of skeletons (in red) generated from CAD models (in grey). The right column represents their corresponding graphs. The blue nodes represent the leaf nodes, yellow denote the non-leaf nodes and edges are represented in red color. The bottom-most representation is on unseen data (armadillo object [33])

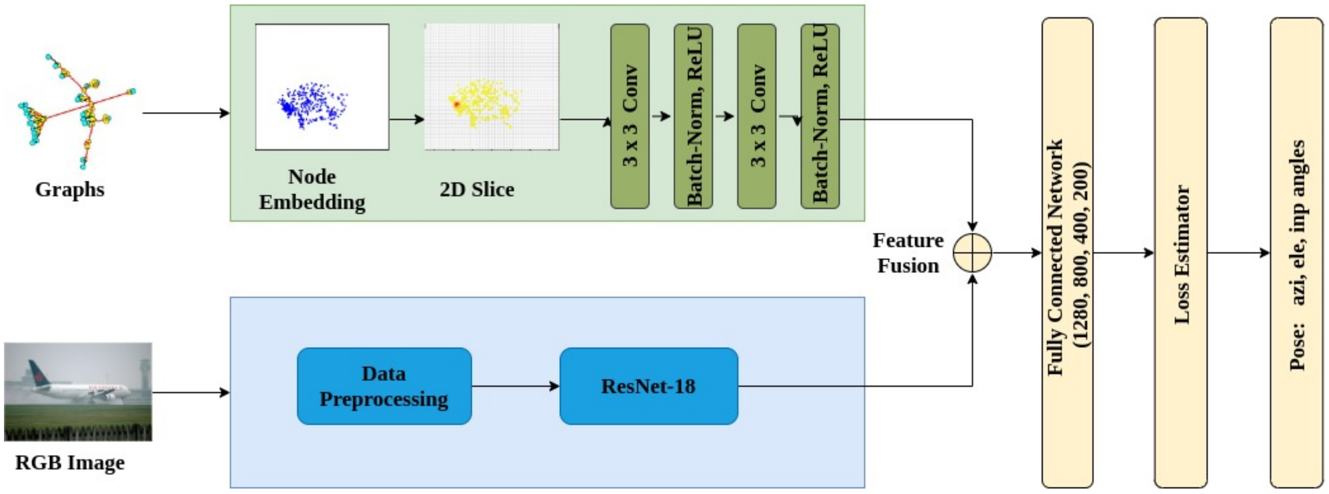

Figure 3: Proposed network architecture: 3DPoseLite exploits 2 parallel pipelines: (a) the green block takes graphs as input, and generates 2D images by extracting slices from compressed representation (obtained via PCA) of node embeddings (via node2vec algorithm [24]); A vanilla CNN later extracts feature vector of dimension 256 while, (b) the blue block takes RGB image as input which after preprocessing is fed to a ResNet-18 model for feature extraction of dimension 1024. The features from pipeline (a) and (b) are fused and passed to a fully connected layer (FCN) and loss estimator for final pose estimation.

3.2. Skeletonization and Graphs

In this regard, we voxelize the meshes of the 3D models and convert them to skeletons via thinning algorithms [27] [31]. The algorithm makes use of *Prairie-Fire analogy*, wherein the boundary of the object is set on fire and the skeleton is the loci where the fire fronts meet and quench each other [9]. These skeletons are transformed to graphs as shown in Figure 2. As compared to Point Cloud and Multi-View [39] [43] representations, this concise, informative, and easily computable graph representation can help reduce the architecture space consumption. The invariance to rotation, translation and scaling make skeleton-based graphs an optimum choice of representation for 3D objects [39], [43]. Even though the number of nodes for each graph is variable (refer supplementary material). But, this variety is in itself a significant feature for the graphs. To be put to use for deep learning models, they need to be processed without any loss of information. Thus, we generate node embeddings which help us pass meaningful information to our neural network.

3.3. Node Embedding Generation

A key bottleneck in applying deep learning or machine learning using graph representation is in transforming the raw graphs into a representation that is easily interpretable by downstream prediction/regression algorithms. To solve this problem, we make use of node embeddings [24]. The objective function of node embeddings is based on the maximum likelihood principle. Specifically, given a graph $G = (V, E)$ with vertices $'V'$ and edges $'E'$, we want to learn a function $f : V \rightarrow R^m$ that maps the vertices to a lower dimensional

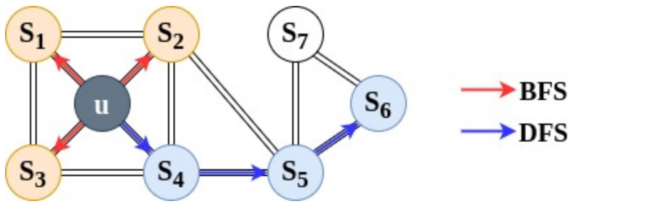

Figure 4: Sampling method using BFS and DFS for neighborhood size = 3 nodes (best viewed in color) [Source: [24]]

space of continuous real valued features. The network neighborhood of each node $'V'$ is computed through a neighborhood *sampling strategy S*. The algorithm *node2vec* [24] tries to optimize the objective function which works on maximizing the log-probability of observing network neighbourhoods conditioned on its feature representation.

Unlike other language processing problems, we need to define a notion of network neighborhood in order to generate the network/graph. These neighborhoods can have vastly different structures based on the sampling strategy S. Figure 4 explains the multiple ways to sample neighbourhood of size = 3 nodes. Breadth First Search (BFS; denoted in red) to find neighborhood with 1 hop is one of the extreme case, while Depth First Search (DFS; denoted in blue) is another. Both are important and thus, `node2vec` algorithm strategically interpolates between BFS and DFS using parameters p and q [24]. A BFS traversal (prioritized by p) captures the communities while a DFS (prioritized by q) discovers structural similarities by traversing farther in the network.

We use 6 combinations of (p, q) in our experiments for

Table 1: Category-wise details on graph nodes and edges for Pascal3D dataset. We report minimum (min), maximum (max) and average of all graphs in a category.

Category	No. of Nodes			No. of Edges		
	Min	Max	Avg	Min	Max	Avg
Aeroplane	32	127	85	34	164	91
Bicycle	69	308	157	76	391	173
Boat	18	103	54	17	110	56
Bottle	2	131	22	1	174	26
Bus	18	53	35	12	62	37
Car	62	391	178	96	587	244
Chair	9	117	43	8	116	47
DTable	2	14	8	1	13	7
MBike	75	770	517	89	973	647
Sofa	14	117	52	15	116	51
Train	23	412	142	20	579	186
TV	2	82	24	1	71	20

augmentation of the node embeddings generated, which is discussed further in Section 3.6. Graph kernels suffer from high time complexity and local optimum problem. Instead of relying on graph kernels to extract features from the computed graphs, we rely on node embeddings which generate much richer information to aid the functioning of our pipeline. We then convert these embeddings into 2D image-like structures so that instead of using complex Graph Cconvolutional Neural Network (GCNNs), a vanilla 2D CNN can do the job. By this approach, we benefit in terms of both time and space complexity. A graph kernel that requires computation of similarity between two graphs of size N performs $N(N-1)/2$ operations. The time complexity of the shortest path graph kernel is $\mathscr{O}(\mid V_1 \mid^2 \mid V_2 \mid^2)$ for two graphs $(V1, V2)$, where $\mid V_i \mid$is the number of nodes in graph V_i. Whereas, generating node embeddings using random walk similarity function [24] has a linear time complexity $\mathscr{O}(E)$ where E is the set of edges in the graph generated. Converting all these graphs to the same dimension and then to 2D images on the dataset level is an $\mathscr{O}(N)$, N being the number of graphs in the dataset.

Thus, we represent graphs as *image-like* structures [44] in three steps; (1) generation of node-embeddings for the graph, (2) space compression obtained using PCA and (3) continuous extraction of 2D slices from the compressed representation and generation of 2D histograms for each slice. Finally, the stack of these 2D histograms is considered as an image, wherein each histogram makes for a channel. The variation in number of nodes is also handled via this approach. The dimensionality of final representation of graph does not depend on its number of nodes/ size of graph. The images generated are fed to custom CNN discussed in next section.

3.4. Feature Extraction and Fusion

We make use of two separate feature extractors; one to encode the shape and one for RGB image. First, for 2D histograms that are generated from graphs, we train a 2 layer CNN (two Convolution2D blocks with Batch Normalization and ReLU activation function followed by dropout with rate = 0.2 (determined empirically varying the drop out parameters) to extract 1D feature vector of dimension 256. Second, for RGB images, we utilize ResNet-18 [26], a standard CNN model to obtain features (a vector of dimension 1024). Later, in order to make use of both 3D shape and RGB images for pose estimation we fuse the features from both and concatenate in an early fusion scheme [13]. Concatenation of vectors provide complementary information obtained from networks and thus help in better pose estimation.

We experimented with many feature fusion techniques including addition, multiplication and weighted feature fusion. We observed a degradation in performance with these. The experiments to change ResNet to Dilated ResNet-22 (DRN-D-22) [51] gives an almost similar accuracy. DRNs are said to preserve spatial information, by removing the average pool layer in the network. For the histograms, we experimented with Inception-V1 module [42]. However, it failed to improve our result and even made the network more complex which is particularly not desirable as we need a fast real-time processing to enhance seamless user interaction.

3.5. Loss Estimator

Our network combines the outputs from pose regression and pose classification i.e., offsets and probabilities respectively. The final loss obtained $\mathscr{L}_{\text{CR}}$, is a summation of probabilities and offsets. For classification, we make use of the standard Cross Entropy Loss $\mathscr{L}_{\text{CE}}$ and Smooth $L1$ loss (Huber Loss) for regression $\mathscr{L}_{\text{L1}}$ [50].

The combined flattened feature vector from both image and graph pipelines is passed to a Fully Connected Network (FCN) for pose estimation with each layer (1280-800-400-200) followed by Batch Normalisation, and ReLU activation. Finally, we obtain the output pose of object with respect to the RGB image in the reference frame in terms of azimuth (az), elevation (el), and in-plane rotation (ip) angle. Each angle $\phi \in \{az, el, ip\}$ is divided into b bins uniformly such that our model outputs l labels $\in$ (0,b$-$1) and δ offsets $\in [-1, 1]$. Thus, our training data consists of 3 inputs to the network (u_i, v_i, p_i), where, u_i is input image, v_i is input histogram of 3D model and p_i is 3 degrees of freedom (DOF) pose given in the annotation file. The Euler angles p_i are converted to bin labels l encoded as one-hot vectors and relative offsets δ within the bins. Loss function $\mathscr{L}_{\text{CR}}$ we use is as below:

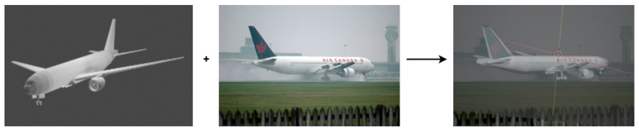

Figure 5: Result of pose estimation on airplane on [48] Dataset

$$\mathscr{L}_{\text{CR}} = \sum_{i=1}^{N} \sum_{\phi} \mathscr{L}_{\text{CE}}(l_{i,\phi}, prob_{\phi}(u_i, v_i)) + \mathscr{L}_{\text{L1}}(\delta_{i,\phi}, reg_{\phi,l_{i,\phi}}(u_i, v_i)) \quad (1)$$

where $prob_{\phi}(u_i, v_i))$ refer to the predicted probabilities for parameterised by ϕ w.r.t. the input image and shape representation followed and $reg_{\phi,l_{i,\phi}}(u_i, v_i))$ correspond to the regressor predicted offset.

3.6. Data Augmentation

The dataset Pascal3D has 79 CAD models for the RGB images. This is a small number to train a neural network and thus we intended to augment these 3D models. We perform augmentation of both images and graphs in our approach. For input RGB images, we make use of standard protocol that includes random jittering of bounding boxes, horizontal rotation/flips and color jittering. In case of node embeddings, we use 6 combinations of parameters (p, q) i.e., $(0.25, 4), (0.5, 2), (1, 1), (1, 0.5), (2, 0.5), (4, 0.25)$ inspired by the work done by Tixier et. al [44]. This design considers all the possible random walks and also the community level features and sub-structural similarity.

3.7. Implementation details

The voxelization, skeletonization and graph retrieval is performed using [27] [31]. The adjacency matrix and edge weights are obtained which is fed to `node2vec` [24] algorithm. Principal Component Analysis (PCA) is performed on the embeddings and histograms (2D images) are obtained. These histograms and input RGB images are fed to the CNN architecture as shown in Figure 3. We divide the entire dataset into train and test as per the protocol followed by [23], [38], [45]. The train dataset consists of ImageNet-trainval and Pascal-train images, while the test data contains 2,113 non-occluded and non-truncated objects of the Pascal-val images (see Sec. 4.1 for additional details on datasets used).

4. Performance Evaluation

4.1. Dataset

For the purpose of training our pose estimator we use the state-of-the-art dataset Pascal3D+ [48], which consists of images with annotations for 12 day-to-day life object categories including aeroplane, bicycle, boat, bottle, bus, car, chair, dining table, motorbike, sofa, train, and tv-monitor. The dataset is compilation of training and validation set images from PASCAL VOC 2012 [15] and ImageNet [14]. The corresponding 3D CAD model is available for each object.

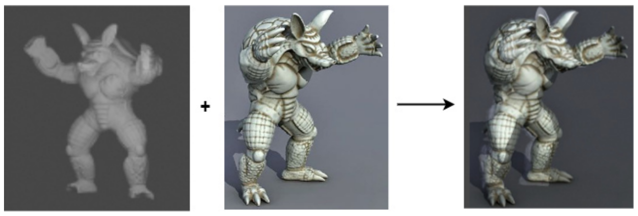

Figure 6: A sample illustration of pose estimation using 3DPoseLite on unseen data, in our case, Armadillo

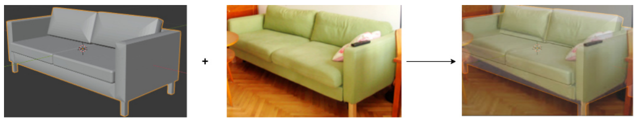

Figure 7: Pose estimation on Pix3D dataset using 3DPoseLite

4.2. Baseline and Evaluation Metrics

Xiao et. al introduced *PoseFromShape* [50] that employ random Multi-Views and PointClouds and feed it to a ResNet model [26] for pose estimation. They claim to achieve accuracy of 82.66% on the Pascal3D dataset as mentioned earlier. For accuracy on Pascal3D dataset, we regard *PoseFromShape* [50] as the baseline for comparison with our model which in turn exploits the potential of graphs to represent 3D shapes effectively. We report our results in terms of accuracy denoted by $Accuracy_{\pi/6}$ where $\pi/6$ is the threshold value. The estimated pose is within this threshold angle.

4.3. Results And Discussion

We first evaluate our method in case the categories of tested objects are covered by training data. We use Pascal 3D+ dataset [48], which comprise of the largest variety of object categories, 3D models and images. We report the results in Table 3. The baseline [50] we follow leverages 3D model of the object and achieves a clear boost in the performance with accuracy 82.66%. Following the same protocol as backbone and introducing graphs in our proposed method, we achieve a comparable accuracy of 80.18%, with computation time reduced by 3× approximately and memory requirement for computation reduced by a factor of 11 as shown in Table 2. The number of parameters required by the model for graph feature extraction is 738,112 (15× less than the baseline) while, earlier for multi-view or pointcloud it required 11,307,840 parameters. The range and average

Table 2: Details on computational efficiency, memory footprint, and number of parameters: Note the reduction in inference time and GPU memory required.

	Train Time	**Test Time** (per instance)	**GPU Memory Required**	**No. of Parameters** (total)
PoseFromShape [50]	112.75 hrs	0.43 sec	8172 MB	23,009,664
3DPoseLite	10.25 hrs	0.15 sec	728 MB	12,439,936

Table 3: Performance Comparison in terms of $Accuracy_{\pi/6}$ on Pascal3D+ [48] dataset. * Not trained on ImageNet data but trained on ShapeNet Renderings

Algorithm	Cat-agnostic	aero	cycle	boat	bottle	bus	car	chair	dtable	mbike	sofa	train	tv	*mean*	*mean*
		$Accuracy_{\pi/6}$													$MedErr$
Tulsiani et. al [45]	$\times$	81	77	59	93	98	89	80	62	88	82	80	80	80.75	13.6
Su et. al* [40]	$\times$	74	83	52	91	91	88	86	73	78	90	86	92	82	11.7
Mousavian et. al [38]	$\times$	78	83	57	93	94	90	80	68	86	82	82	85	81.03	11.1
Grabner et. al [23]	$\surd$	80	82	57	90	97	94	72	67	90	80	82	85	81.33	11.5
PoseFromShape [50]	$\surd$	81	83	60	93	97	91	79	67	90	90	81	79	82.66	10
3DPoseLite	$\surd$	80	82	58	93	96	92	77	57	88	82	80	79	80.18	13.4

number of nodes and edges are reported in Table 1 category-wise. The number of nodes is significantly less as compared to point clouds and thus can serve as the reason for the drop in number of parameters, making our deep learning model lighter and portable to devices. Note that the accuracy could be further improved by hyper-parameter tuning on our approach. These results mostly exceed or are on par with category-specific approaches discussed [45] [40], [29]. Some of the models developed [29] make use of select category of objects instead of entire dataset, making them biased. Even though underlying idea is competitive, they fail to generalize on a wide variety of objects in our day-to-day life.

We show the result on an RGB image from the dataset in Figure 5. The pose estimated is approximately accurate and subsequently the registration is bound to be accurate owing to accurate pose estimation. To illustrate the generality on unseen data, we show a sample on a set of Armadillo images from Stanford 3D scanning repository [33] on which the 3DPoseLite (our network) is not trained on and our framework tends to learn the underlying structure. The graph generated (as shown in Figure 2) is meaningfully captures the significant points including well defined nodes for head, ears, hands and legs. Adding to it, the pose estimation (as shown in Figure 6) is quite accurate even when we consider completely different category of objects (i.e., an animal unlike rigid body objects). Also, Table 4 and Figure 7 shows the testing performance comparison of proposed method on Pix3D dataset [41]. This depicts the robustness of graph generation procedure that encodes the shape accurately and also the pipeline developed.

But realizing the scope of improvements, we analyze some possible failure cases of our method. Situations where (i) images have multiple objects of similar category; (ii) objects get occluded (or camouflaged) by background and are hard to detect; (iii) the pose is such that part of geometry of object is not visible. In spite of these limitations, the desirable characteristics make our neural network model suitable for on-device registration tasks with low latency and speed-up performance. This is crucial for internal overlays or 3D model registration to an image seamlessly in a resource constrained environment.

5. Future Work and Conclusion

We have presented a new compact neural network architecture for deep pose estimation termed 3DPoseLite. We demonstrated the benefits of this approach in terms of computational efficiency and, memory footprint, with a negligible trade-off in accuracy. Our proposed method 3DPoseLite achieves comparable results to the state-of-the-art on standard pose estimation datasets. We highlight and demonstrate that our approach is promising for pose estimation on generic unseen objects without any re-training and, in some sense, our approach is unsupervised in nature.

To facilitate the compactness of neural networks, we make use of node embeddings to represent 3D shapes as

Table 4: Performance Comparison in terms of $Accuracy_{\pi/6}$. Trained on Pascal3D+ [48] dataset and tested on Pix3D dataset [41]

Algorithm	tool	misc	bcase	wdrobe	desk	bed	table	sofa	chair	*mean*
	$Accuracy_{\pi/6}$									
Georgakis et. al [45]	-	-	-	-	34.90	50.80	-	-	31.20	-
PoseFromShape [50]	10.90	13.10	22.30	6.60	52.00	55.30	35.60	64.60	35.80	32.90
3DPoseLite	8.70	10.00	62.31	57.23	66.45	57.86	39.97	94.05	50.10	45.75

graphs. Instead of passing raw data-driven learning to extract features and graph kernels which are both costly and time-consuming, we train our model on relevant features extracted via a simple 2-layer CNN. We demonstrate it via benchmarking on standard datasets comprising of rigid body objects. This work pushes forward the state-of-the-art and we see the potential of utilizing graphs in pose estimation for rigid body objects.

In the future, we intend on extending our work to compress neural networks via pruning graphs and efficient representations in feature extractor modules. For our research work to see the light of the day in actual pose estimation problems and internal overlays where a 3D object registration to a live feed is essential, a compact model is the need of the hour. We aim to achieve this through additional techniques using off-the-shelf model compression techniques such as ONNX [7] and TensorFlowLite frameworks which will aid in the development of real-time applications with a better user experience for Augmented Reality.

References

[1] Arkit. `https://developer.apple.com/documentation/arkit`. Accessed: 2020-09-28.

[2] Augmented pro. `https://www.augmentedpro.com/`. Accessed: 2020-06-11.

[3] Diotasoft. `http://www.diotasoft.com/index.php/en/`. Accessed: 2020-06-11.

[4] Metaio. `https://en.wikipedia.org/wiki/Metaio`. Accessed: 2020-06-11.

[5] Model target generator. `https://library.vuforia.com/articles/Solution/model-target-generator-user-guide.html`. Accessed: 2020-06-11.

[6] Wikitude. `https://www.wikitude.com/documentation/`. Accessed: 2020-06-11.

[7] J. Bai, F. Lu, K. Zhang, et al. Onnx: Open neural network exchange. *GitHub repository*, 2019.

[8] P J Besl and N D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[9] H Blum et al. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380, 1967.

[10] Anna Brounstein, Ilker Hacihaliloglu, Pierre Guy, Antony Hodgson, and Rafeef Abugharbieh. Towards real-time 3d us to ct bone image registration using phase and curvature feature based gmm matching. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 235–242. Springer, 2011.

[11] Tat-Jun Chin, Zhipeng Cai, and Frank Neumann. Robust fitting in computer vision: Easy or hard? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 701–716, 2018.

[12] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. A deep convolutional activation feature for generic visual recognition. *UC Berkeley & ICSI, Berkeley, CA, USA*.

[13] Ramya ebbalaguppe, Kevin McGuinness, Jogile Kuklyte, Rami Albatal, Cem Direkoglu, and Noel E O'Connor. Reduction of false alarms triggered by spiders/cobwebs in surveillance camera networks. In *ICIP*, pages 943–947. IEEE, 2016.

[14] Deng et al. Imagenet: A large-scale hierarchical image database. In *2009 IEEE CVPR*, pages 248–255. Ieee, 2009.

[15] Everingham et al. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[16] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[17] Luis Ferraz, Xavier Binefa, and Francesc Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–508, 2014.

[18] M A Fischler and R C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[19] K Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.*, 36(4):193–202, 1980.

[20] Georgios Georgakis, Srikrishna Karanam, Ziyan Wu, and Jana Kosecka. Learning local rgb-to-cad correspondences for object pose estimation. In *Proceedings of the IEEE Inter-*

national Conference on Computer Vision, pages 8967–8976, 2019.

[21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[22] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 437–446, 2015.

[23] A Grabner, P M Roth, and V Lepetit. 3d pose estimation and 3d model retrieval for objects in the wild. In *Proceedings of the IEEE CVPR*, pages 3022–3031, 2018.

[24] A Grover and J Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on KDD*, pages 855–864, 2016.

[25] F Hagelskjær and A. G. Buch. Pointposenet: Accurate object detection and 6 dof pose estimation in point clouds. *arXiv preprint arXiv:1912.09057*, 2019.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Philip Kollmannsberger, Michael Kerschnitzki, Felix Repp, Wolfgang Wagermaier, Richard Weinkamer, and Peter Fratzl. The small world of osteocytes: connectomics of the lacuno-canalicular network in bone. *New Journal of Physics*, 19(7):073019, 2017.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[29] Jogendra Nath Kundu, Aditya Ganeshan, Rahul MV, and Aditya Prakash. ispa-net: Iterative semantic pose alignment network. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 967–975, 2018.

[30] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[31] Ta-Chih Lee, Rangasami L Kashyap, and Chong-Nam Chu. Building skeleton models via 3-d medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

[32] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[33] J. Gerth B. Curless Levoy, Marc and K. Pull. The stanford 3d scanning repository. *URL http://www-graphics. stanford. edu/data/3dscanrep*, 5:7, 2005.

[34] E Marchand, H Uchiyama, and F Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE T VIS COMPUT GR*, 22(12):2633–2651, 2015.

[35] Primoz Markelj, Dejan Tomaževič, Bostjan Likar, and Franjo Pernuš. A review of 3d/2d registration methods for image-guided interventions. *Medical image analysis*, 16(3):642–661, 2012.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[37] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[38] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[39] Mattia Natali, Silvia Biasotti, Giuseppe Patanè, and Bianca Falcidieno. Graph-based representations of point clouds. *Graphical Models*, 73(5):151–164, 2011.

[40] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. 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*, pages 2686–2694, 2015.

[41] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2974–2983, 2018.

[42] Wei Liu Yangqing Jia Pierre Sermanet Scott Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Szegedy, Christian and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[43] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 3d skeletons: A state-of-the-art report. In *Computer Graphics Forum*, volume 35, pages 573–597. Wiley Online Library, 2016.

[44] Antoine J-P Tixier, Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Graph classification with 2d convolutional neural networks. In *International Conference on Artificial Neural Networks*, pages 578–593. Springer, 2019.

[45] S Tulsiani and J Malik. Viewpoints and keypoints. In *Proceedings of the IEEE CVPR*, pages 1510–1519, 2015.

[46] Rui Wang, Chenyang Huang, and Xiangyang Wang. Global relation reasoning graph convolutional networks for human pose estimation. *IEEE Access*, 8:38472–38480, 2020.

[47] Jimmy Wu, Bolei Zhou, Rebecca Russell, Vincent Kee, Syler Wagner, Mitchell Hebert, Antonio Torralba, and David MS Johnson. Real-time object pose estimation with pose interpreter networks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6798–6805. IEEE, 2018.

[48] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE winter conference on applications of computer vision*, pages 75–82. IEEE, 2014.

[49] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[50] Yang Xiao, Xuchong Qiu, Pierre-Alain Langlois, Mathieu Aubry, and Renaud Marlet. Pose from shape: Deep pose estimation for arbitrary 3d objects. *arXiv preprint arXiv:1906.05105*, 2019.

[51] F Yu, V Koltun, and T Funkhouser. Dilated residual networks. In *Proceedings of the IEEE CVPR*, pages 472–480, 2017.

[52] Hong Zhang, Hao Ouyang, Shu Liu, Xiaojuan Qi, Xiaoyong Shen, Ruigang Yang, and Jiaya Jia. Human pose estimation with spatial contextual information. *arXiv preprint arXiv:1901.01760*, 2019.

[53] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. Semantic graph convolutional networks for 3d human pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3425–3435, 2019.