RoMaViD: Learning Robotic Manipulation from Video Demonstrations

Abhinav Upadhyay
Accenture Labs
Bangalore, India
k.a.abhinav@accenture.com

Alpana Dubey
Accenture Labs
Bangalore, India
alpana.a.dubey@accenture.com

Shubhashis Sengupta
Accenture Labs
Bangalore, India
shubhashis.sengupta@accenture.com

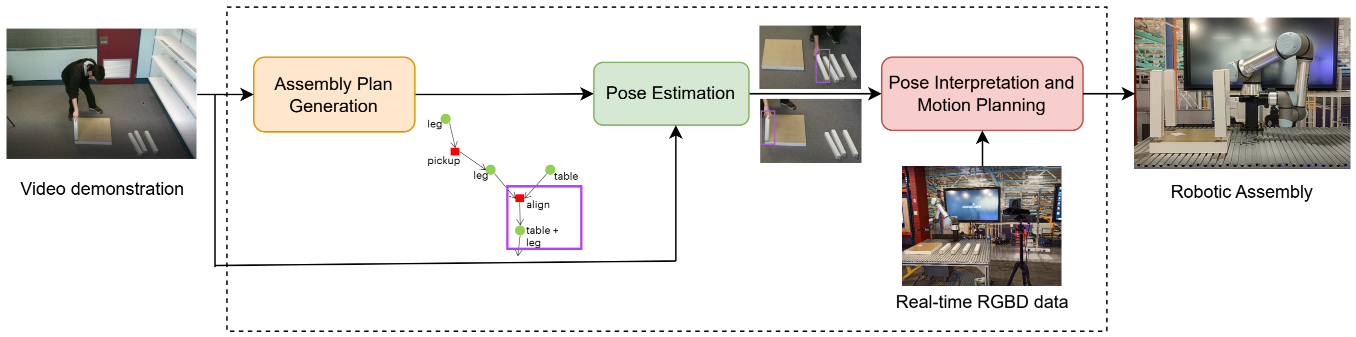


Figure 1: Framework for Robotic Manipulation through Video Demonstrations

ABSTRACT

Large-scale manufacturing necessitates automation, and robotic automation has emerged as a primary solution. Traditionally, robotic systems are designed for fixed assembly lines dedicated to specific product sets. However, with an increasing demand for specialized and customized products, there is a growing need for more agile manufacturing processes. To address this, we introduce a robotic assembly framework capable of generating assembly plans directly from RGB-D video demonstrations. We develop a pose estimation model to capture changes in object poses. We demonstrate its effectiveness in the robotic assembly of IKEA furniture, emphasizing its precision in managing assembly tasks.

CCS CONCEPTS

• **Computing methodologies** → **Robotic planning**; **Vision for robotics**; *Activity recognition and understanding*.

KEYWORDS

Robotic Assembly, Learning from Demonstration, Plan Generation

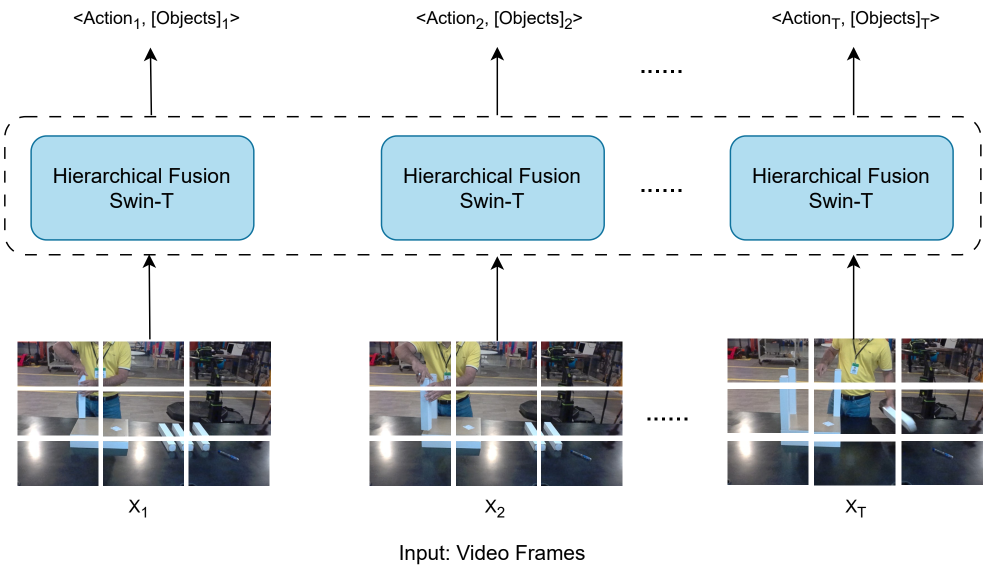
ACM Reference Format:
Abhinav Upadhyay, Alpana Dubey, and Shubhashis Sengupta. 2024. RoMaViD: Learning Robotic Manipulation from Video Demonstrations. In *2024 ACM/IEEE 6th International Workshop on Robotics Software Engineering (RoSE '24), April 15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3643663.3643966

1 INTRODUCTION

In the rapidly advancing landscape of robotics systems, particularly in the manufacturing sector, there has been a significant surge in the utilization of robots for intricate tasks. The manufacturing industry, in particular, has harnessed the potential of robots extensively to execute a myriad of repetitive functions, with a significant surge in the deployment of arm robots for assembly tasks. While these achievements mark substantial progress, the current emphasis has shifted towards achieving true agility in robotic assembly, especially considering that motion and manipulation planning for robots is often pre-determined for a fixed set of product assemblies, particularly in cellular manufacturing. There is a need for autonomous robotic assembly systems that can dynamically adapt to different products.

In this work, we present a robotic assembly framework that leverages RGB-D video demonstrations for product assembly planning. Our novel approach identifies objects and actions from assembly videos, enabling the generation of effective assembly plans. We predict the 6-DoF pose of objects with respect to the video frame and transform them to pose with respect to the experimental set-up frame. We use our framework to demonstrate the robotic assembly of IKEA furniture. Our contributions can be summarised as follows:

- We propose a novel Hierarchical Swin Transformer network for the identification of both objects and actions from a video demonstration.
- We propose a 6-DoF pose estimation model to track the change in the pose of objects in the video.
- A pose interpretation method to transform the pose values from the video frame to the experimental set-up frame.



<Action$_1$, [Objects]$_1$>
<Action$_2$, [Objects]$_2$>
<Action$_T$, [Objects]$_T$>
Hierarchical Fusion Swin-T
Hierarchical Fusion Swin-T
Hierarchical Fusion Swin-T
X_1
X_2
X_T
Input: Video Frames

Figure 2: Hierarchical Swin Transformer network to predict the actions performed & identify the different objects in the video.

2 RELATED WORK

Robotic assembly is an active area of research. Many studies have been conducted for performing autonomous robotic assembly. These studies involve methods for extracting and representing assembly details from sources such as videos, graphical manuals or CAD files [17]. Zakka et al. [21] use a reinforcement learning approach for the robotic assembly of kits. The approach utilizes three modules – the suction module to predict picking locations, the placing module to predict placing locations and the matching module to match objects to their placing locations. Paulius et al. [12] propose Functional Object-Oriented Network (FOON) to represent human activities and manipulations. They manually construct these functional units by labeling cooking instructional videos. Haage et al. [5] propose an interface for robotic assembly from RGB-D videos. Object detection and pose estimation of the hands are performed to generate keyframes. Wan et al. [19] propose a human teaching and a robot execution phase. In the human teaching phase, object detection is followed by reading AR markers to determine the pose. The robot execution phase involves object detection, pose estimation, heuristic-based grasp planning, and motion planning. Sera et al. [16] use graphical instruction manuals for robotic assembly and perform task planning by generating Assembly Task Sequence Graphs (ATSG). The ATSG is a directed graph representing the assembly procedure through the actions involved between objects.

Our approach is different from the aforementioned works along two aspects

(1) Our novel approach identifies objects and actions from complex assembly video demonstrations to generate assembly plans. Our approach can effectively generate complex assembly plans for multiple parts interactions.
(2) Our approach does not require April tags [19][8] or manually provided pose values [16] for detecting the 6-DoF pose of objects in a video or real-world frame. We propose a deep neural network to predict the 6-DoF pose of objects in complex and cluttered scenes and even in occluded scenarios.

3 APPROACH

In this section, we describe our robotic manipulation framework (as shown in Figure 1). The framework consists of three modules:

(1) **Assembly plan generation:** For each frame in the video, we detect the objects and the actions performed on them. Then, we generate a assembly plan representing the objects and actions. The assembly plan is a directed graph with two types of nodes - object nodes and action nodes.
(2) **Pose estimation:** We perform the 6-DoF pose estimation for the objects in the video. This module provides the pose of objects in their assembled state and the pose changes involved in reaching that state.
(3) **Pose interpretation and motion planning:** The pose values obtained from the previous module are with respect to the video frame. These values are transformed to our experimental set-up frame using real-time RGB-D data. The robot replicates the assembly using the assembly plan and the transformed pose values. Motion planning is performed to navigate the experimental set-up.

3.1 Assembly Plan Generation

Initially, we discuss the network designed for action prediction and object identification within video frames. Subsequently, we outline an approach for generating assembly plan based on the identified actions and objects.

We present a novel Hierarchical Fusion Swin Transformer network designed for action prediction and object identification from video demonstrations (as shown in Figure 2). Our contribution lies

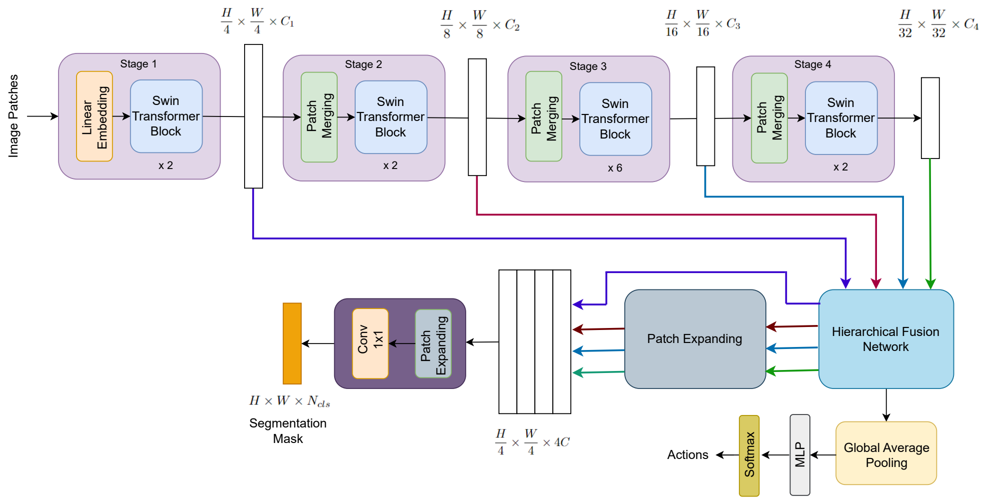


Figure 3: Hierarchical Fusion Swin-T network

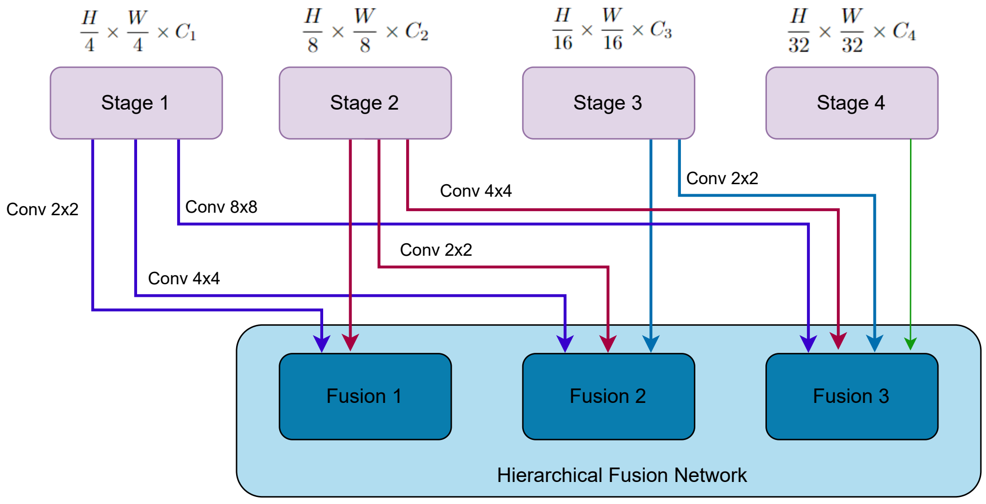

$\frac{H}{4} \times \frac{W}{4} \times C_1$
$\frac{H}{8} \times \frac{W}{8} \times C_2$
$\frac{H}{16} \times \frac{W}{16} \times C_3$
$\frac{H}{32} \times \frac{W}{32} \times C_4$
Stage 1
Stage 2
Stage 3
Stage 4
Conv 2x2
Conv 8x8
Conv 4x4
Conv 4x4
Conv 2x2
Conv 2x2
Fusion 1
Fusion 2
Fusion 3
Hierarchical Fusion Network

Figure 4: A Hierarchical Feature Fusion module integrated after each Swin Transformer block, to enhance the extraction of both local features and global correlations. This module applies convolutions to feature maps generated by preceding stages.

in the introduction of a hierarchical feature fusion module, facilitating the aggregation of features at various stages. We represent video V as a sequence of frames, denoted as $V = \{X_1, X_2, ..., X_T\}$. Each input frame is divided into fixed-sized, non-overlapping patches ($4 * 4$ in our case) using a patch-splitting module. Each patch is considered as a "token," and its feature representation is generated by combining pixel values and passing the raw features to a linear embedding layer, which transforms them into a fixed dimension. The Swin Transformer block (as shown in Figure 3) analyzes these patches, extracting multi-level features at resolutions corresponding to 1/4, 1/8, 1/16, and 1/32 of the original image. Downsampling and upsampling operations on the feature maps are carried out by patch merging and patch expanding blocks, respectively. In the Swin Transformer, patch merging takes place at each stage, bringing together feature maps from the previous stage. This process involves grouping neighboring patches to create larger spatial regions, generating new feature maps that are both spatially accurate and semantically enriched.

The Swin Transformer model faces a significant limitation in terms of inter-stage interaction among feature maps with different scales or resolutions [11]. The lack of a direct communication mechanism hinders the model's ability to capture global context and dependencies across images. To address this constraint, our proposed solution introduces a novel Hierarchical Feature Fusion Module (as shown in Figure 4) after each Swin Transformer block. This module aims to enhance the extraction of both local features and global correlations by strategically convoluting feature maps from preceding stages. At each stage, except Stage 1, we aggregate multi-level features by applying convolution to feature maps from the current stage and the preceding stages. For example, in Stage 4,

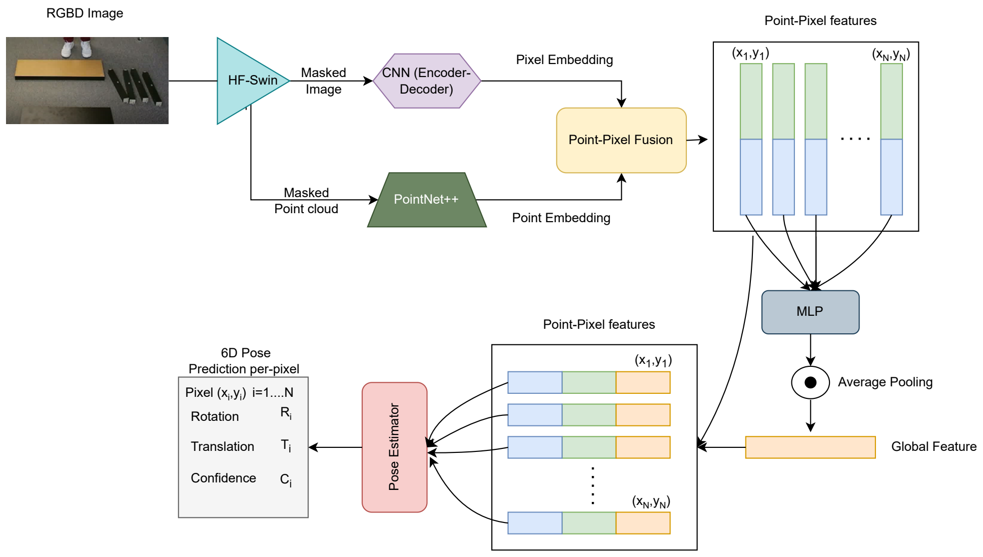


Figure 5: Pose estimation network

convolution is applied to feature maps from Stage 1 (Conv $8 * 8$), Stage 2 (Conv $4 * 4$), and Stage 3 (Conv $2 * 2$). Subsequently, these feature maps are combined with the Stage 4 feature map. The Hierarchical Fusion Network Block acts as a channel to facilitate the exchange of information between feature maps of varying resolutions. This approach ensures the combination of multi-level feature maps from each stage, resulting in the generation of spatially and semantically enriched information.

The segmentation mask is generated by passing the concatenated feature to a sequence of operations, which includes an MLP layer, a Patch Expanding block, and a Convolutional block. The multi-level features obtained from the fusion module undergo channel dimension unification through an MLP layer. Subsequently, these features are upsampled to 1/4th of their original size using a Patch Expanding block and then combined. Finally, a convolutional layer with a 1x1 kernel processes the fused feature to predict the segmentation mask, denoted as M, with a resolution of $\frac{H}{4}$ x $\frac{W}{4}$ x N_{cls}, where N_{cls} represents the number of segmentation classes.

Generating Assembly Plan: To generate the assembly plan, we leverage the concept of functional object-oriented network (FOON) [12]. The Functional Object-Oriented Network (FOON) is a conceptual framework designed to represent and organize activities within a task, particularly in the context of assembly planning. It takes the form of a directed graph, comprising two primary node types: object nodes and action nodes. Object nodes represent entities involved in activities, such as leg, side panel, shelf, and drawer, while action nodes represent specific actions that can be performed on the objects, such as attach, align, rotate, and connect. The FOON's structure facilitates the depiction of the sequential flow of activities, with certain nodes being outcomes resulting from the interaction between others. Edges, denoted as E, connect nodes, signifying the relationships and dependencies between objects and actions in the assembly process.

The assembly graph generation algorithm comprises the following steps:

(1) The model predicts actions and identifies objects for every frame within the video.
(2) The generated information is structured and saved in a JSON file, encapsulating the actions and objects corresponding to each frame.
(3) The algorithm iterates through frames in the video, extracting predicted actions and objects for each frame from the JSON file while eliminating redundant outputs observed across multiple frames.
(4) Employing the Functional Object-Oriented Network (FOON) concept, the algorithm constructs a graph characterized by a bipartite structure.

3.2 Pose Estimation

Our goal is to estimate the 6-DoF pose of objects present in video demonstrations and then transform the pose of the object with reference to the camera coordinate frame in the experimental setup. Formally, a 6-DoF pose p is represented as a rotation R and a translation t, $p = [R|t]$. As we estimate the 6-DoF pose of the objects from the frames in the video, the poses are defined with respect to the camera coordinate frame. The network for pose estimation from a video is shown in Figure 5. Our pose estimation network is inspired from [20] with two changes to the network - (a) Processing of 3D point cloud object is implemented using modified PointNet++ [13] rather than PointNet, and (b) Implemented fusion strategy for segmented point cloud and image map.

The network takes an RGB-D image as an input and generates segmentation masks for each object category. Then, the segmented mask is passed through the encoder-decoder network to generate pixel embedding. For each object, we feed the 3D point cloud (obtained from depth pixels) to PointNet++ [13] network to generate point (geometry) embedding. The pixel and point embeddings are fused at each pixel level using the Point-Pixel fusion network. The pose predictor module predicts 6-DoF pose values along with the confidence score using point-pixel features. We discuss these modules in detail.

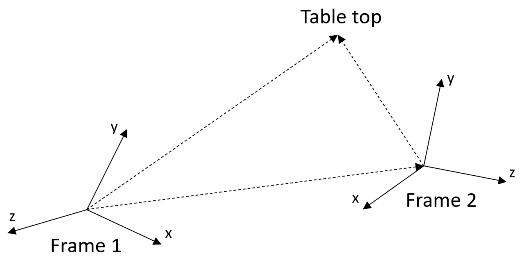


Figure 6: The position of the table top with respect to the video demonstration (frame 1) and the experimental set-up (frame 2).

3.2.1 Object Segmentation. We use Hierarchical Fusion Swin Transformer network to segment different objects in the scene. For details, please refer Section 3.1.

3.2.2 Geometry (Point) Embedding - PointNet++. We propose a variant of PointNet++ [13] as a geometric embedding network that generates a per-point feature by mapping each of the n segmented points to a d_{geo}-dimensional feature space. The network maps a point cloud object $n \times 3$ into a $n \times d_{geo}$ (where n is the number of points in the segmented point cloud object and d_{geo} is the embedding size, 1024 in our case) embedding space. PointNet++ has proven to be effective for 3D classification and segmentation [13]. PointNet++ uses multi-scale grouping (MSG) [13] that learns multi-scale features using multiple groups of weights. To capture more efficient geometric relations, we apply three-scale neighborhoods centered on a sampled point with a shared weight.

3.2.3 Pixel (RGB) Embedding. We use a CNN-based Encoder-decoder architecture as a pixel (RGB) embedding network that generates a per-pixel feature by mapping an image of the size of H x W x 3 to H x W x d_{pi} (where d_{pi} is the embedding size, 1024 in our case) embedding space.

3.2.4 Point-Pixel Fusion. The pixel and point embeddings obtained from the image and the 3D point cloud respectively need to fuse effectively to represent the feature space to handle occlusion and imperfect segmentation [20][7]. Point-Pixel fusion consists of two components - Pixel-to-point fusion and Point-to-pixel fusion.

Pixel-to-point fusion combines RGB (pixel) features to point cloud features. For each point feature, we find its K-nearest points in the pixel map and their corresponding feature embeddings. These feature embeddings are then passed through max pooling and a shared MLP to obtain the pixel feature.

$$F_{pi2p} = MLP(max(F_{pi}); \forall K \in K_{pi}) \tag{1}$$

where F_{pi} are the feature embeddings of the K-nearest points in a pixel (RGB) map.

Finally, a shared MLP is applied to the concatenation of pixel and point features to obtain the fused point feature (F_{fp}).

$$F_{fp} = MLP(F_p \oplus F_{pi2p}) \tag{2}$$

Similar to pixel-to-point fusion, the point-to-pixel fusion modules obtain fused pixel features (F_{fpi}).

$$F_{p2pi} = MLP(max(F_p); \forall K \in K_p) \tag{3}$$

where F_p are the feature embeddings of the K-nearest points in a point (geometry) map.

$$F_{fpi} = MLP(F_{pi} \oplus F_{p2pi}) \tag{4}$$

The point-pixel fusion features are then passed through MLP and pooling layers to generate a fixed-size global feature vector. The global feature vector is concatenated with each of the point-pixel features to provide global context. The resulting point-pixel features are then passed through MLP layers to predict a set of N poses, one for each point-pixel feature, along with a confidence score c_i for each prediction.

Training Objective: We apply point-pixel loss [20] as the training objective, which measures the distance between the points sampled on the objects model in the ground truth pose and the corresponding points on the same model transformed by the predicted pose.

3.3 Pose Interpretation and Motion Planning

The pose values observed in the assembly plan are predicted by the pose estimation network with respect to the camera frame of the video demonstration (frame 1). These values cannot be directly used to replicate the pose change for the camera frame of our experimental set-up (frame 2). For the experiment, we assume that the initial position of the table top with respect to the real-world frame has remained unchanged. The camera has undergone a positional displacement that has resulted in the two frames as shown in Figure 6. This assumption is used to calculate the transformation matrix, which in turn used to obtain the pose values with respect to our experimental set-up.

In Figure 6, the pose of the table top with respect to frame 1 can be denoted as t^1, R^1 where t^1 is the 3D translation and R^1 is the rotation matrix defining the orientation. t^2, R^2 is the pose with respect to frame 2. The relative 3D translation and orientation of frame 2 with respect to frame 1 is calculated [15] using

$$t^1 = {t_2}^1 + t^2 \tag{5}$$

$$R^1 = {R_2}^1 R^2 \tag{6}$$

t^1 was obtained from the assembly plan. R^1 was calculated using the quaternion values from the assembly plan. t^2 and R^2 for the experimental set-up were calculated using the known camera parameters.

For frame 2 with respect to frame 1, the transformation matrix can be denoted in terms of the rotation matrix and 3D translation [15] as

$${T_2}^1 = \begin{bmatrix} {R_2}^1 & {t_2}^1 \\ 000 & 1 \end{bmatrix} \tag{7}$$

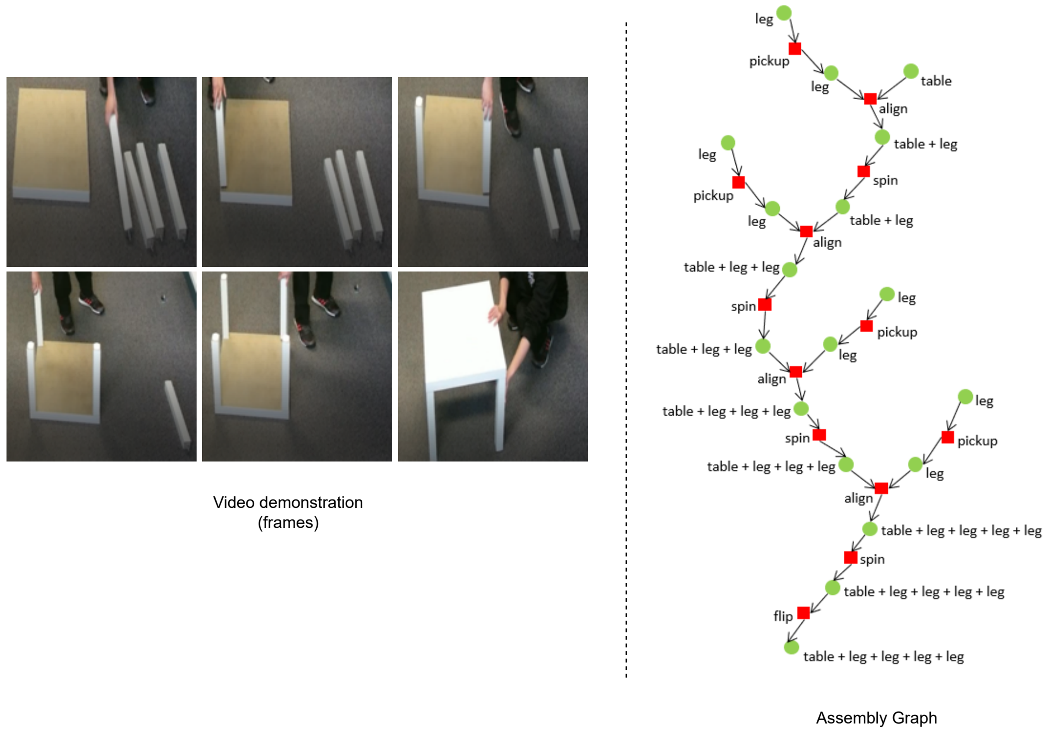


Figure 7: The assembly plan generated for an IKEA side table (partly shown here).

The pose values P^2 for the experimental set-up, are calculated from the pose values P^1 in the assembly plan using the transformation matrix ${T_2}^1$ as

$$P^1 = {T_2}^1 P^2 \tag{8}$$

Only the optimal pose of the assembled parts from the assembly plan is transformed to the experimental set-up frame.

We use RRT connect [9] for motion planning.

4 EVALUATION AND IMPLEMENTATION

4.1 Dataset

The IKEA ASM dataset comprises 371 distinct assemblies featuring four furniture types: side table, coffee table, TV bench, and drawer, accompanied by pose information [1]. In total, there are 1113 RGB videos and 371 depth videos (top view). The dataset encompasses 3,046,977 frames (~35.27h) of footage, averaging 2735.2 frames per video (~1.89min). With a total of 16,764 annotated actions, the dataset exhibits an average of 150 frames per action (~6sec).

4.2 Evaluation

4.2.1 Action Recognition. We assess our action recognition approach using three metrics [1]: (i) **Frame-wise accuracy (FA)**: It represents the fraction of correctly classified frames over the total frames in each video, averaged across all videos in the test set, (ii) **Macro-recall**: Due to the dataset's imbalance, we compute macro-recall by calculating recall for each category separately and then averaging it., and (iii) **Mean average precision (mAP)**: Since all videos have multiple action labels, we compute the mean Average Precision, which is calculated by taking the mean Average Precision

Table 1: Performance of our approach along three metrics and comparison with baseline

Approaches	Frame accuracy		M-recall	mAP
	Top 1	Top 3		
ResNet18 [1]	27.06	55.14	21.95	11.69
ResNet50 [6][1]	30.38	56.1	20.03	9.47
C3D [18][1]	45.73	69.56	32.48	21.98
I3D [2][1]	57.57	76.55	39.34	28.59
P3D [14][1]	60.4	81.07	45.21	29.86
TSM [10]	63.52	84.2	50.36	34.72
Ours	**75.18**	**92.41**	**69.52**	**56.34**

Table 2: Performance of our segmentation approach (on the testing dataset) along two metrics and comparison with baselines

Approach	Mean Accuracy	Mean IoU
PSPNet [22]	0.63	0.6
DeepLabV3+ [3]	0.72	0.68
Swin-T [11]	0.81	0.72
Ours	**0.9**	**0.83**

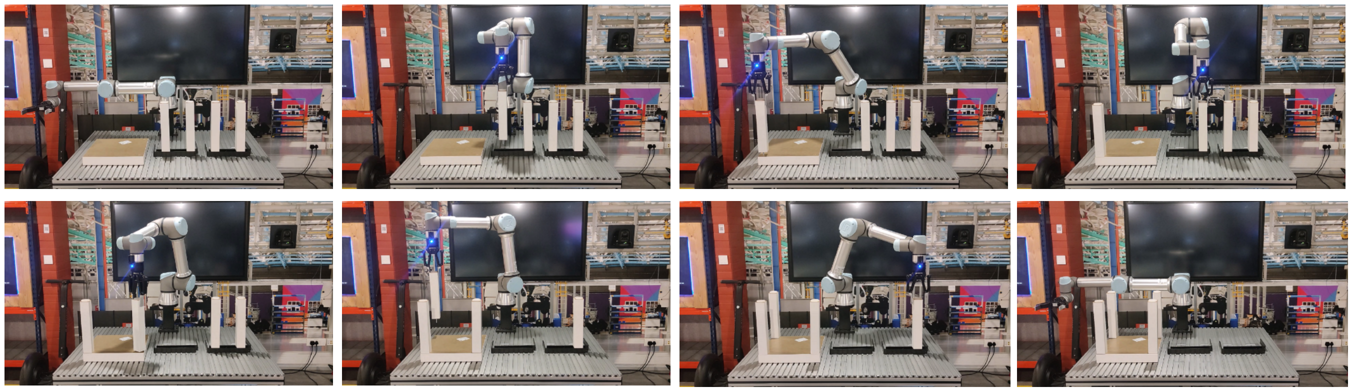


Figure 8: Robotic execution of assembly of an IKEA table. The sub-figures correspond to different steps of the Assembly plan (due to space constraints, only a subset of the steps are shown here).

Table 3: Quantitative evaluation of 6-DoF pose on IKEA ASM dataset. Metrics capture the mean score along all the objects.

Approaches	AUC	ADD-S<1cm
PoseCNN [4]	82.2	83.1
DesneFusion [20]	86.3	87.8
Ours	**91.2**	**93.4**

over all classes. Table 1 presents the performance of our approach based on these metrics on the IKEA ASM [1] dataset.

4.2.2 Part Segmentation. We assess the effectiveness of our segmentation approach using two key metrics: (a) Mean accuracy, which represents the proportion of correctly classified pixels averaged across all classes, and (b) Mean intersection over union (IoU), indicating the overlap area between predicted and ground truth pixels averaged over classes. The performance of our segmentation approach is presented in Table 2 with respect to these metrics.

We compare our approach with existing baselines, namely PSPNet [22], DeepLabv3+ [3], and Swin-T [11], across these two metrics. Our results demonstrate a notable improvement, with a 11% increase in Mean Accuracy and an 15% increase in Mean IoU compared to [11].

The assembly plan generated for an IKEA side table is shown in Figure 7.

4.2.3 Pose Estimation. We use two metrics [20] to evaluate our pose estimation approach - (a) ADD-S curve (AUC): ADD-S computes the mean distance from each 3D model point transformed by predicted pose $[\hat{R}|\hat{t}]$ to its closest neighbor on the target model transformed by ground truth pose $[R|t]$. We report the area under the ADD-S curve (AUC) and set the maximum threshold of AUC to be 0.1m. (b) ADD-S < $1cm$: We report the percentage of ADD-S smaller than $1cm$, which measures the predictions under the minimum tolerance for robot manipulation (1cm for most of the robot grippers). Table 3 shows the performance of our pose estimation approach.

4.3 Implementation

The robotic assembly framework assembles an IKEA side table (35cm x 35cm) using a UR5e arm robot equipped with a Robotiq 2F-140 gripper. Real-time RGB and depth data are obtained through a Microsoft Kinect v1, positioned 1 meter from the bottom-right corner of the setup, facing the workbench. The depth data is crucial for extracting pose and size information of items on the table and for collision avoidance.

In the implementation, depicted in Figure 8, the UR5e robot starts in the zero-position, with all joint positions set to 0 and the gripper fully open. IKEA furniture parts are randomly placed on the table, and their poses and dimensions are computed using real-time depth data. Assembly actions are iteratively extracted from the assembly plan (stored as a JSON file). The robot then executes these actions on the parts.

5 CONCLUSION

In this work, we present a novel approach, the Hierarchical Swin Transformer network, designed to predict actions and identify objects within videos demonstrating furniture assembly. We employ the Functional Object-Oriented Network concept to automatically generate assembly plans. Additionally, we develop a pose estimation model to accurately capture variations in the objects' poses. Our framework is utilized to demonstrate the robotic assembly of IKEA furniture, and the results underscore the efficacy of our proposed approach in addressing robotic assembly tasks. Furthermore, our method can be easily generalized for other manipulation tasks, such as disassembly, troubleshooting, etc.

REFERENCES

[1] Yizhak Ben-Shabat, Xin Yu, Fatemeh Saleh, Dylan Campbell, Cristian Rodriguez-Opazo, Hongdong Li, and Stephen Gould. 2021. The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision.* 847–859.

[2] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 6299–6308.

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV).* 801–818.

[4] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. 2015. P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE international conference on computer vision*. 3218–3226.

[5] Mathias Haage, Grigoris Piperagkas, Christos Papadopoulos, Ioannis Mariolis, Jacek Malec, Yasemin Bekiroglu, Mikael Hedelind, and Dimitrios Tzovaras. 2017. Teaching assembly by demonstration using advanced human robot interaction and a knowledge integration framework. *Procedia Manufacturing* 11 (2017), 164–173.

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

[7] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. 2021. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3003–3013.

[8] Jonathan D Jones, Cathryn Cortesa, Amy Shelton, Barbara Landau, Sanjeev Khudanpur, and Gregory D Hager. 2021. Fine-grained activity recognition for assembly videos. *IEEE Robotics and Automation Letters* 6, 2 (2021), 3728–3735.

[9] James J Kuffner and Steven M LaValle. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, Vol. 2. IEEE, 995–1001.

[10] Ji Lin, Chuang Gan, and Song Han. 1811. Temporal shift module for efficient video understanding. CoRR abs/1811.08383 (2018).

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[12] David Paulius, Yongqiang Huang, Roger Milton, William D Buchanan, Jeanine Sam, and Yu Sun. 2016. Functional object-oriented network for manipulation learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2655–2662.

[13] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* 30 (2017).

[14] Zhaofan Qiu, Ting Yao, and Tao Mei. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*. 5533–5541.

[15] Alasdair Renfrew. 2004. Introduction to robotics: Mechanics and control. *International Journal of Electrical Engineering & Education* 41, 4 (2004), 388.

[16] Issei Sera, Natsuki Yamanobe, Ixchel G Ramirez-Alpizar, Zhenting Wang, Weiwei Wan, and Kensuke Harada. 2021. Assembly planning by recognizing a graphical instruction manual. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3138–3145.

[17] Garrett Thomas, Melissa Chien, Aviv Tamar, Juan Aparicio Ojea, and Pieter Abbeel. 2018. Learning robotic assembly from cad. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3524–3531.

[18] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 4489–4497.

[19] Weiwei Wan, Feng Lu, Zepei Wu, and Kensuke Harada. 2017. Teaching robots to do object assembly using multi-modal 3d vision. *Neurocomputing* 259 (2017), 85–93.

[20] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. 2019. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 3343–3352.

[21] Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. 2020. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 9404–9410.

[22] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2881–2890.