2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR) | 979-8-3503-7202-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/AIxVR59861.2024.00023

Equipment Assembly Recognition for Augmented Reality Guidance

Kevin Murray*†, Jonathan Schierl*, Kevin Foley*, Zoran Duric‡
*Overlab LLC, Clifton, VA
kevin.murray@theoverlab.com
†Department of Computer Science
George Mason University, Fairfax, VA
zduric@gmu.edu

***Abstract*—Equipment maintenance is a challenging task, particularly for complex equipment with many parts. Augmented Reality (AR) technology can assist technicians by providing real-time, on-site guidance. A fundamental requirement for this guidance is recognizing the current pose and assembly state of the equipment. This work addresses the problem of recognizing the pose and assembly state of equipment from multiple visible light images, specifically aiming to handle real-world equipment with hundreds of parts and millions of possible assembly states. We propose a novel two-stage method that first estimates a coarse pose and assembly state, then refines these estimates by leveraging multi-view integration of 2D features in a 3D voxel grid. Our approach is validated on two real assemblies with hundreds of parts: a small engine and a 3D printer. Experimental results demonstrate the effectiveness of our method, with refinement improving both pose and assembly state estimates. This work contributes a new perspective to AR-guided equipment maintenance, highlighting the importance of valid assembly states in training and the benefits of multi-view feature integration for assembly recognition.**

***Index Terms*—augmented reality, assembly, pose, multi-view, synthetic data**

I. Introduction

Equipment assembly recognition is of significant value in many industries, including manufacturing, maintenance, and inspection. As an example, augmented reality (AR) can assist a technician in performing maintenance tasks on equipment by overlaying step-by-step instructions, including which parts to remove and in which order. To provide these instructions with holographic markers from the user's perspective, the AR system must estimate the pose of the assembly. To guide the user on the next steps in assembly/disassembly, it must also estimate the assembly state, i.e., which parts are currently present or absent.

We thus define *assembly recognition* as the two part problem of:

1) *6D pose estimation* (comprising 3D position and 3D rotation)
2) *Assembly state estimation* (the N-dimensional binary vector representing the presence or absence of N parts)

This project was sponsored by NAVAIR under Contract N6833521C0507, and it has been cleared for public release under NAVAIR Public Release 2023-644 Distribution Statement A, which states: 'Approved for public release; distribution is unlimited.'

In this paper, we address assembly recognition using multiple images captured from an AR headset as a user moves around the equipment. This is a challenging problem due to the vast variety of equipment types, the large number of parts that can be added or removed in an assembly, the complexity of real-world environments, and the need to operate on embedded hardware.

Our approach to this problem is driven by the following key contributions:

- We focus on real, complex assemblies with hundreds of parts, millions of possible assembly states, unknown equipment orientations, and cluttered environments. The two test assemblies are shown in Fig. 1.
- We restrict ourselves to 2D, visible light images. We do this because we have found that depth cameras do not work well on the shiny/dark surfaces that are often encountered with real equipment, as others [1] have also found.
- We define and take advantage of "valid" assembly states, using an undirected graph to represent the connections between parts in the assembly.
- We extend the common correspondence estimation and render-and-compare 6D pose estimation techniques to the assembly recognition problem.
- We introduce a feature-level, multi-view refinement stage that outperforms the common render-and-compare approach in this problem.

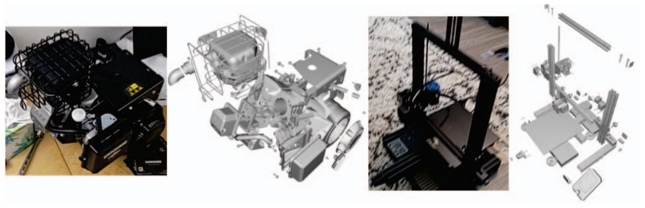

Fig. 1. Images and exploded assemblies of the engine (left) and 3D printer (right) test objects.

These contributions allow us to recognize the pose and assembly state of real, complex equipment in real-world environments, significantly extending the state of the art in assembly recognition.

A. Problem Definition

Before delving into our methodology, it is crucial to define the specific problem we are addressing. We focus on assembly recognition, where an assembly is composed of multiple rigid parts, each represented as a mesh with its own coordinate system within a common assembly reference frame. Each part is characterized by a specified pose, which includes both translation and rotation with respect to the assembly reference frame. We expect to have such an assembly model in advance, e.g. from a Computer-Aided Design (CAD) model.

An assembly can be represented as an undirected graph, where the vertices are parts and the edges represent the connections between parts. A simple, four part assembly is shown in Fig. 2. We define a valid assembly state as one that can be treated as a single rigid object, i.e., not multiple separate bodies that can move freely with respect to each other. Thus, a valid assembly state is a connected subgraph of the full assembly graph. Referring again to Fig. 2, the connected subgraph of $\{bolt1, band1\}$ would be valid, while the unconnected subgraph of $\{bolt1, bolt2\}$ would be invalid, since those two parts alone would be unconstrained.

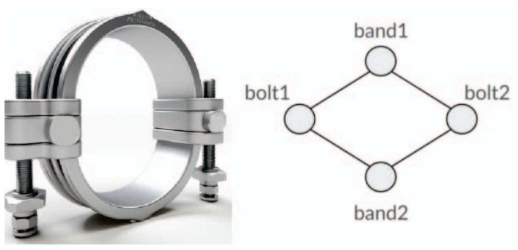

Fig. 2. A pipe clamp and associated assembly graph.

An assembly state, A, specifies the presence or absence of the individual parts and is represented as a binary vector: $A \in \{0, 1\}^N$, where N is the number of parts. The goal of assembly recognition is to estimate both the pose of the assembly and the assembly state.

Our problem comes with several assumptions and specifications, which we detail below:

- **Valid Assembly State**: We assume that the equipment is in a valid assembly state, which we define as a collection of parts forming a single, rigid object, rather than a collection of scattered parts. This assumption aligns with practical scenarios where equipment maintainers are working with assembled or partially assembled equipment.
- **Multiple Views of Known Camera Poses**: We work with multiple views of known camera poses, a common situation with AR headsets, which track the headset pose as the user moves. This allows us to capture different perspectives of the assembly, aiding in the recognition process.
- **Visible Light Images**: Our approach uses visible light images. While depth images are often used in computer vision tasks, they tend not to work well with many types of real equipment due to issues like shiny surfaces. Visible light images, on the other hand, provide a more reliable source of information for our task.
- **No Real Labeled Data for Training**: We focus on the practical case where no real labeled images are available for training our networks. In lieu of real data, we use synthetic, ray-traced images for network training. This scenario is common, as acquiring a large number of labeled real-world images is often difficult and time-consuming.
- **Unknown Texture/Material**: We do not rely on textures or materials because many existing equipment CAD models do not include them.
- **Unknown Equipment Orientations**: The equipment may be rested at unknown orientations, introducing additional complexity to the task of assembly recognition.
- **Realistic Environment Challenges**: We operate in environments with imperfect lighting, cluttered backgrounds, and occlusions. These conditions reflect the realistic challenges faced in practical applications of AR in equipment maintenance.

By defining these assumptions and specifications, we set the stage for a method that is practical, robust, and applicable to a wide range of real-world equipment maintenance scenarios.

II. Related Work

A. AR-Assisted Assembly Guidance

Augmented Reality (AR) has emerged as a powerful tool in assembly processes, particularly in guiding users through complex tasks. Research, such as that presented in [2], has demonstrated the potential of AR to significantly aid users in assembly operations. The effectiveness of AR in improving accuracy and reducing assembly time is well documented, showcasing its utility in industrial and manufacturing settings.

However, a common approach in AR-assisted assembly is the use of fiducial markers to facilitate object tracking. While effective, this method faces several challenges. As noted in the survey paper [3], marker-based tracking is prevalent in AR assembly applications, but these systems often struggle with issues such as marker occlusion and difficulties in attaching markers to small industrial components. These limitations can lead to incorrect registration in the AR environment, potentially diminishing the effectiveness of the AR assistance.

Despite these challenges, the literature consistently points to the potential effectiveness of AR in assembly guidance. The integration of AR into assembly processes has been shown to provide tangible benefits, but the reliance on fiducial markers remains a significant hurdle. This reliance on markers and the associated challenges underscore the need for innovative approaches in AR-assisted assembly, particularly methods that can operate effectively without markers.

B. Assembly Recognition

Assembly recognition is a complex task that has seen limited exploration. In [4], the assembly state estimation problem is addressed, but the scope is confined to assemblies with only six states. Crucially, this work does not include assembly pose estimation, which is essential for accurately referencing specific locations on the assembly.

More comprehensive assembly recognition systems, which consider both assembly state and pose, include the marker-less methods proposed by [5] and [6]. These methods, focusing on segmentation and corner feature extraction, have shown promise within their specific applications. However, their applicability is limited to simpler assemblies, with a linear sequence of up to 11 assembly states and a presupposed vertical orientation. Additionally, these methods were evaluated in controlled laboratory settings and depend on depth information, which is problematic on shiny or dark surfaces frequently encountered in industrial equipment.

In contrast, our work addresses these limitations by focusing on complex real-world assemblies with millions of possible assembly states, unknown equipment orientation, cluttered environments, and operates solely on visible light images.

C. 6D Pose Estimation

The problem of 6D pose estimation, often applied in robot bin picking, involves determining the six degrees of freedom (3 translations and 3 rotations) that describe an object's pose in space. The BOP Challenge [7], a competition focused on benchmarking 6D pose estimation methods, has largely driven recent research in this area.

The methodologies in this field generally follow a multi-stage process: detection, coarse pose estimation, and pose refinement. Recent methods have predominantly leveraged deep learning [7] and are trained on high-fidelity synthetic data generated through ray tracing, using tools like BlenderProc [8].

In the first stage, object bounding boxes are identified with a detector. This stage can be seen as a limited coarse pose estimator that is refined in the second stage. Knowledge of the image position and extents from the detector allows the coarse pose estimator to train on cropped images with canonical scales. This is similar to our refinement approach that performs a 3D crop of the 3D, multi-view scene.

Most coarse pose estimators focus on segmenting the object and estimating object correspondences, using a convolutional neural network (CNN) [9]. These correspondences can be found by directly regressing object coordinates [10], or by forming features that can be matched with features from the mesh [11]. The pose is then estimated from these correspondences using the perspective-n-point (PnP) algorithm.

Refinement is typically done using a render-and-compare (RAC) approach. This method involves generating a predicted image by rendering the object mesh at the coarse pose estimate, comparing that to the actual image, and computing a pose offset. This approach was introduced by [12] and expanded on by [13]. Although effective, it has some limitations. The refinement process forms a new image and uses a new CNN, so it cannot make use of the features from the coarse pose CNN. For some complex assemblies, rendering may be a significant computational cost, especially on embedded hardware. Lastly, it is inherently a single view procedure.

While the BOP Challenge and related methods have made significant progress in single-image pose estimation, we believe assembly recognition essentially requires multiple views due to parts being occluded. Furthermore, multiple views with known poses are commonly available in commercial AR headsets. While [13] introduces a multi-view refinement strategy, it requires multiple objects in the scene and does not take advantage of known camera poses. In contrast, our work takes advantage of known camera poses and integrates multiple views at the feature level.

D. Multi-View Reconstruction

Multi-view reconstruction involves using multiple images of a scene, taken from different viewpoints, to reconstruct a 3D model of the scene. In this problem, the camera poses are typically known for each image, allowing information across views to be aggregated at the pixel or feature level.

A key work in this field is [14], which uses a 2D CNN to extract features from each image, and then back-projects the multi-view features into a common 3D voxel grid. This voxel grid is then processed by a 3D CNN. The idea of back-projecting 2D CNN features into a 3D voxel grid is valuable and something we make use of in our refinement stage.

While these methods are largely successful in reconstructing full 3D scenes, they do not directly address the task of recognizing specific objects or assembly states within those scenes. Inspired by [14], our work focuses on the specific task of assembly recognition and introduces a novel approach that effectively leverages the 3D voxel grid to improve pose and assembly state estimation.

In conclusion, our work extends existing literature in assembly recognition, 6D pose estimation, and multi-view reconstruction by introducing a method that handles the complexities of real-world assemblies.

III. Methodology

Our solution to the assembly recognition problem consists of three key components:

1) Synthetic data generation
2) Coarse assembly recognition
3) Pose and assembly state refinement

We do not use a detector as a first stage because we are generally dealing with one large object close to the camera, rather than several small objects in the same scene (which is the typical situation for robot bin picking).

All of the neural networks in our method are trained exclusively on the synthetic data (Section III-A). This is a practical necessity, as collecting and labeling data in a multitude of assembly states would be extremely time consuming.

Our coarse estimation procedure (Section III-B) is based on a common approach for 6D pose estimation, but extended to assemblies and multiple views.

We introduce two alternative methods for assembly refinement that improve both the pose estimate and the assembly state classifications. One extends the render-and-compare technique (Section III-C) and one uses a canonical voxel grid to integrate multi-view features (Section III-D) All three methods are illustrated in Fig. 3.

A. Synthetic Data Generation

The purpose of synthetic data generation is to create a large-scale, realistic dataset of the assembly in various poses and valid assembly states for training our networks.

To generate the dataset, we first represent the assembly as an undirected graph, where vertices represent the parts and edges indicate the connections between the parts. These connections are inferred based on locality. Specifically, we use the PyBullet [15] mesh collider to detect if two meshes are in contact, in which case we create an edge between the corresponding vertices in the graph.

Random, valid assembly states are generated by sampling connected subgraphs of the assembly graph. For each valid assembly state, we procedurally generate a multi-view scene using NVISII [16]. The scene includes a floor plane onto which the assembly and several background meshes are dropped using the PyBullet [15] physics engine, resulting in natural orientations.

To account for the variability in real-world conditions, we apply the concept of domain randomization [17]. This involves randomizing several aspects of the scene, such as the positions of light sources, camera poses, textures/materials, skyboxes, and light sources.

For each scene, we generate multiple images along with ground truth labels that include equipment segmentations, equipment point correspondences, camera poses and intrinsics, equipment pose, and assembly state. Fig. 4 shows example synthetic images of the engine assembly.

B. Coarse Assembly Recognition

Our coarse estimator (Fig. 3, orange) is an extension of [10], an encoder-decoder U-Net [18] CNN that segments objects and regresses object coordinates per pixel. In [10], the pose is obtained by estimating correspondences between 3D object coordinates and 2D image pixels.

To extend [10] to multiple views, we also estimate the depth to equipment pixels. This allows us to convert the 2D image pixels into 3D camera coordinates. These camera coordinates are converted into world coordinates, using the known camera poses.

Thus, we obtain a set of 3D-3D correspondences between the equipment coordinate system and the world coordinate system, across all camera views. The transformation between the equipment and world frames is robustly estimated from the 3D world and 3D equipment correspondences using Horn's method [19], with outlier rejection using random sample consensus (RANSAC) [20].

Assembly state estimation. We estimate the assembly state, $A \in \{0,1\}^N$, which specifies the presence or absence of the N individual parts. This is achieved using a multi-label binary classifier positioned after the encoder stage of the CNN. The classifier outputs the confidence of each of the N parts being present.

At test time, these part-level confidences are obtained for each image and then averaged across all images to yield the final assembly state estimate. We employ averaging under the assumption that part-level confidences are approximately independent across different images, acknowledging that this is a reasonable but not strictly true approximation. The averaging enhances the robustness of the state estimation.

C. Render-and-Compare Refinement

Our first pose and assembly state refinement technique involves a comparison of the actual and predicted images. As with other RAC approaches, the ”predicted” images are obtained by rendering the object mesh(es) according to a coarse prediction, and the ”actual” images are the images obtained from the camera.

Our RAC network (Fig. 3, blue) is trained to compute an offset between both the predicted and actual poses, and an offset between the predicted and actual assembly states. The assembly state offset is a binary vector representing whether there is a difference between the predicted and actual image, with respect to each part. For example, if both images show part n is missing (or both show it to be present), the offset for part n is $false$. If there is a difference, then the offset for part n is $true$.

Like other render-and-compare methods, we use a CNN with a four-channel input: one grayscale channel representing the predicted image and three color channels for the actual image. The predicted image is in grayscale because we assume texture and material information is unknown.

The network has three outputs: a 9D vector encoding the pose offset, a predicted error between the estimated pose offset and the actual pose offset, and a N-D vector representing the offsets for each of the N parts in the assembly. The 9D pose offset is based on the parameterization first suggested by [21] and implemented by [13].

To prepare the images for processing, we first crop them around the assembly. This is done by computing the assembly's extents using the coarse pose estimate. Therefore, this refiner network is always trained and run on cropped images, providing consistent scale.

During testing, we run our coarse estimator to obtain a coarse pose and coarse assembly state. We then render this coarse estimate and run the refiner for each of the multi-view images. The refiner estimates the confidence of a coarse part classification being incorrect, for each part in the assembly. We average these estimates across the input views. If the average confidence is above 0.5, the refiner predicts that the coarse estimate for that part is wrong, and thus inverts it. For our refined pose estimate, we simply use the refined pose from the image with the lowest predicted pose error.

D. Canonical Voxel Grid Refinement

Our second pose and assembly state refinement technique is inspired by multi-view reconstruction, specifically the multi-view 3D CNN developed by [14]. This multi-view approach

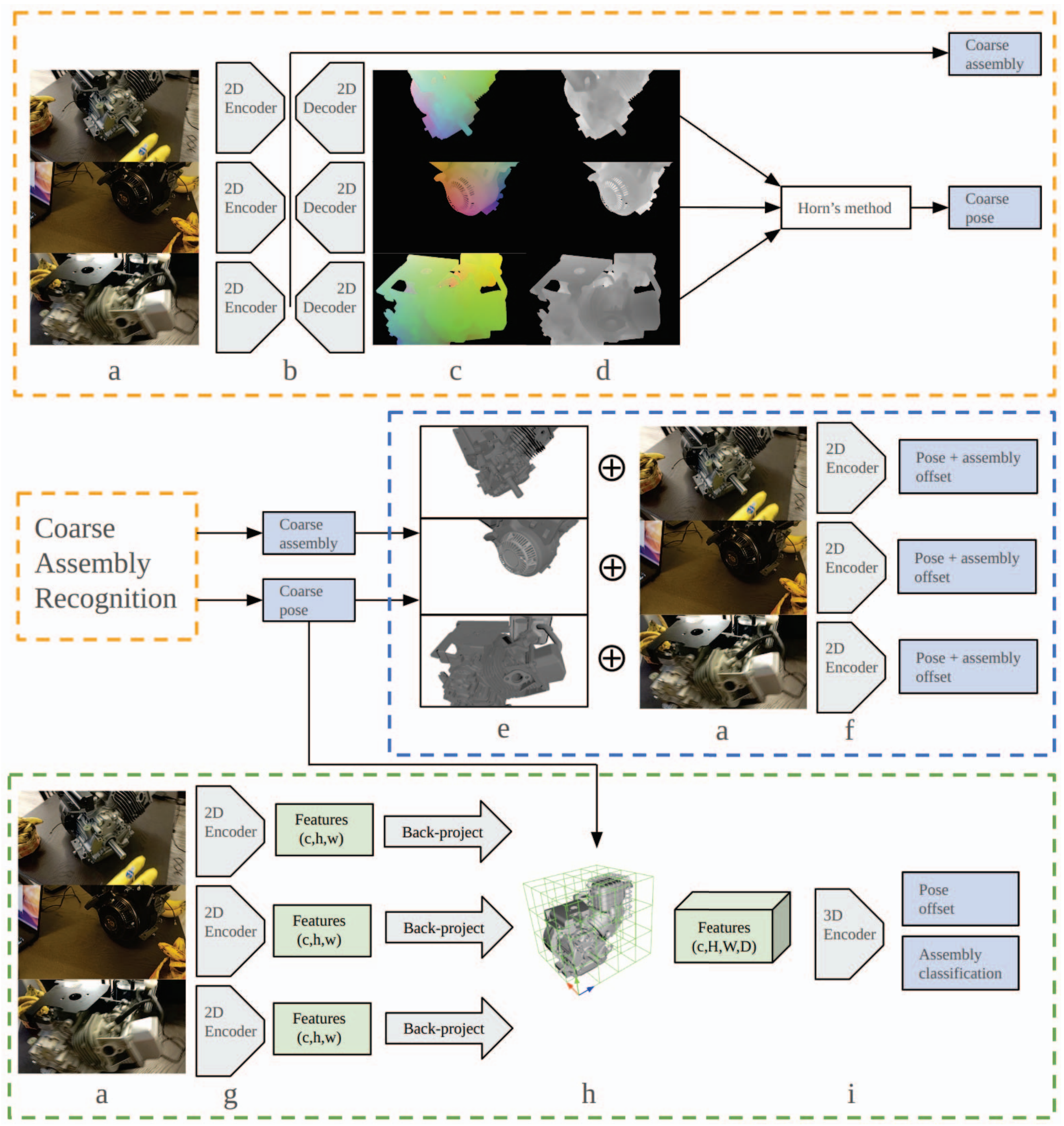

Fig. 3. Full system architecture, including the coarse assembly recognition subsystem, render-and-compare refinement subsystem, and canonical voxel grid refinement. The coarse estimates are input into the two, alternative refinement subsystems. All subsystems are input with multi-view images (a) with known camera projection matrices. **Coarse assembly recognition** (orange): images are input to the U-Net CNN (b). The output shows the segmented equipment point correspondences (c) (x, y, and z coordinates are normalized and visualized in red, green, and blue, respectively) and the segmented depth image (d). For clarity, the ground truth outputs are shown. The assembly state is estimated after the CNN encoder for each image and then averaged across the images. **Render-and-compare refinement** (blue): predicted images (e) are rendered using the assembly mesh, camera projection matrices, and coarse estimates of the assembly state and pose. These are channel-wise concatenated with the corresponding input images and then input to the CNN encoder (f), which estimates offsets for both pose and assembly state. **Canonical voxel grid refinement** (green): a 2D encoder (g) extracts features from each image. The features are back-projected into the canonical voxel grid (h) using the camera projection matrices and coarse pose estimate. The 3D voxel grid is then processed with a 3D encoder (i), which estimates a pose offset and an assembly classification.

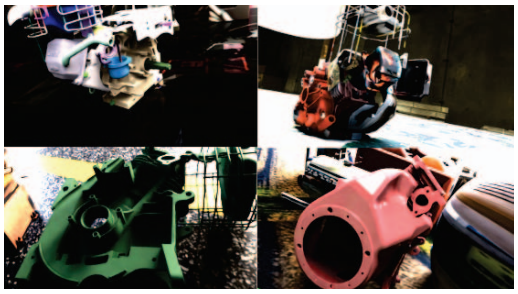

Fig. 4. Synthetic images of the engine assembly.

provides the network with a richer, more complete representation of the scene, which can lead to improved accuracy in the refined pose and assembly state estimates. The process involves five key steps (illustrated in Fig. 3, green):

2D Feature Extraction: A 2D CNN encoder is used to extract features from each of the input images.

Voxel Grid Formation: A voxel grid is defined in world coordinates, sized and oriented to tightly envelope the assembly. The size is defined in advance based on the assembly extents. The pose of the voxel grid is based on the coarse pose estimate of the assembly, which provides positional consistency of features in the voxels (thus a ”canonical” voxel grid).

Multi-view Feature Integration: The 2D CNN features from all input images are back-projected into the voxel grid, following [14]. Briefly, each voxel center, v_{ijk} is projected into image coordinates, using the known camera pose and intrinsics. The 2D CNN feature at that image coordinate is assigned to the voxel. This is repeated for each view, and the final input feature for each voxel is obtained by max-pooling features across views (again, following [14]).

Multi-view Processing: A 3D CNN encoder processes the multi-view voxel grid and outputs a 3D feature map describing the overall scene, near and oriented with the equipment.

Pose and Assembly State Refinement: The last module of this network uses the 3D feature map to regress a pose offset and classify the assembly state $A \in \{0,1\}^N$. We parameterize the pose offset as a 3D translation vector and a 6D rotation, as suggested by [21]. This is the same parameterization of the render-and-compare approach, except that we do not have separate translation units for depth and pixels, since we are operating in a 3D space with consistent dimensions. The pose offset is relative to the canonical pose encoded in the voxel grid. The output pose offset is then applied to the initial pose estimate, yielding a refined pose estimate. The assembly state is estimated similarly to the coarse estimator (Section III-B): a multi-label binary classifier that outputs the confidence of each of the N parts being present.

Although the initial assembly state prediction is not explicitly encoded in the voxel grid, the canonical orientation of the voxel grid allows the network to more effectively recognize different assembly states. The network outputs a binary vector representing the presence or absence of each part in the assembly.

This process effectively allows the network to focus on the task of refining the pose and assembly state estimates, while taking advantage of the full 3D structure of the scene as seen from all available viewpoints.

IV. Experiments

We validate our approach on two challenging test assemblies, namely an engine and a 3D printer, shown in Fig. 1. The engine is a Briggs & Stratton 19L232-0054-G1 10 HP Vanguard Engine with 134 parts and the printer is a Creality Ender3 3D printer with 250 parts. Both assemblies are commercially available with community-created 3D models.

Lincoln-Peterson estimation [22] indicates the number of valid assembly states is two million for the engine and eight million for the printer. Without the ”valid assembly state” constraint, the total number of possible states would be 2^N, where N is the number of parts.

A. Data Collection

We collected real test data using the Microsoft HoloLens 2. For each assembly, we collected 30 scenes with unique assembly states, where on average 44% of the engine’s parts and 40% of the printer’s parts were missing. Each scene consists of a sequence of images captured at 30 Hz as a user naturally moves around the object, typically covering about 180 degrees of azimuth. The scenes are between 3 and 10 seconds in duration. We manually labeled the ground truth assembly pose and assembly state for each scene.

B. Implementation

For efficiency, we limit our inference resolution to 384x224 and process six images from the scene, for all three methods. Accordingly, when generating synthetic data, we render images at 384x224 and render six images per scene. We generate 50,000 synthetic training scenes per assembly, and train all three networks using this data. All three methods utilize a 2D CNN encoder, which we implement with EfficientNet-b0 [23]

Our coarse estimator is based on a U-Net with the EfficientNet-b0 CNN encoder, with the decoder and skip connections implemented with [24]. To limit computation in the transform estimation, we subsample 1,000 output correspondences per image.

The assembly state classifier is implemented with a max pooling layer that condenses the encoder’s feature map into a single representative feature, followed by a fully connected layer that produces N output values, each corresponding to the estimated presence of a part.

Our render-and-compare refiner uses the EfficientNet-b0 CNN encoder, expanded to handle input images with four channels. We resize the cropped image to 256x256, applying letterboxing as necessary. The last layer of features is max-pooled and run through a fully connected layer to estimate the pose offset, predicted error, and assembly state offset.

For training, the ”actual” images in this context are our ray-traced images, while the ”predicted” images are rendered with PyRender [25]. Errors in the predicted images are simulated with random perturbations. We perturb the pose with a 2.5 cm, 2.5 degree standard deviation error, and we perturb the assembly state by randomly inverting the classification of 0-50% of the parts. An example training input is shown in Fig. 5. To train the pose error prediction, we compute the pose error using (1), which we introduce in Section IV-D. We compute a loss for the predicted error using mean square error between the predicted error and actual error.

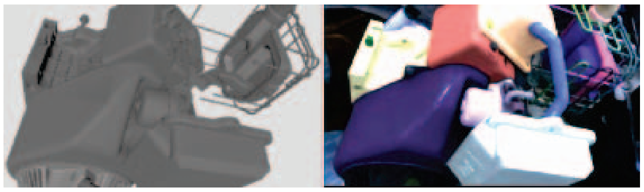
Fig. 5. Example render-and-compare training input for the engine assembly. Note how the predicted image (left) is missing parts that are shown in the actual image (right).

Our canonical voxel grid refiner extracts features from each image with the EfficientNet-b0 CNN. The features are back-projected into a 12x12x12 voxel grid, sized as a cube slightly larger than the target assembly, and positioned based on the coarse pose. During training, we simulate coarse pose estimates by perturbing the pose of the voxel grid with 2.5 cm, 2.5 degree standard deviation error. The 3D CNN is a three layer residual [26] encoder. The pose and assembly state module max-pools the 3D feature map and uses a fully connected layer to obtain the pose offset and assembly state.

We train both the coarse and refiner networks on the synthetic data for 100 epochs using the Adam optimizer with an initial learning rate of 0.0001. Training includes significant data augmentation: Gaussian noise, channel shuffle, color jitter, and motion blur. The latter of which is particularly important due to the limitations of the real HoloLens 2 camera. We avoid any augmentation operations that modify scene geometry.

All three methods use the binary cross-entropy loss function to train assembly classification. The coarse method uses binary cross-entropy for equipment segmentation, mean square error (MSE) for equipment point regression, and L1 loss for depth regression. The RAC refiner and canonical voxel grid refiner use the pose regression loss from [13].

Both refinement methods are tested with real coarse inputs: the coarse pose and assembly estimates we get from the coarse estimator.

C. Hardware and Run Times

All networks were trained and evaluated on a desktop computer equipped with an NVIDIA RTX 3090 GPU, an Intel i9-10850K CPU, and 64 GB of RAM. Table I shows the training and run times for each of the three methods: coarse estimation, render-and-compare refinement (RAC), and canonical voxel grid refinement (Voxel). Each test processes six images at the resolutions described in Section IV-B.

Note that the run time for render-and-compare refinement is largely due to rendering the part meshes from the equipment model. These meshes have been downsampled, but there is likely still room for further optimization.

Currently, only the coarse method has been ported to the HoloLens 2, where its run time is 6.5 seconds (again with six images at 384x224 resolution). This underscores the need for efficiency on embedded AR devices.

The canonical voxel grid refinement method has the potential to be even more efficient by sharing 2D CNN encoder weights with the coarse estimator. Both methods currently use the same encoder architecture (EfficientNet-b0) and run on the same input images, but are trained and inferred separately.

TABLE I
TRAINING AND RUN TIMES

Method	Training Time (hours)	Run Time (ms)
Coarse	34	163
RAC	76	2,600
Voxel	34	107

D. Evaluation

For each test, we randomly select six test images from the full sequence, which allows us to compute several tests per scene. The output of each test is a predicted pose and predicted assembly state, at both the coarse and refined level.

We evaluate our pose estimates by computing the Maximum Surface Distance (MSD) between the estimated and ground truth poses, shown in (1). The MSD represents the maximum distance error of a point, $\mathbf{x}$ on the surface of the equipment, between its transformation from the ground truth pose ($\widehat{\mathbf{P}}$) to the estimated pose ($\widetilde{\mathbf{P}}$).

$$e_{MSD}(\widehat{\mathbf{P}}, \widetilde{\mathbf{P}}, \mathbf{N_M}) = max_{\mathbf{x} \in \mathbf{N_M}} \left\| \widehat{\mathbf{P}}\mathbf{x} - \widetilde{\mathbf{P}}\mathbf{x} \right\|_2 \quad (1)$$

The MSD is based on the standard maximum symmetric surface distance (MSSD) used in the BOP Challenge [27], except that we drop the symmetry term, since our test objects are asymmetric.

We evaluate our assembly state estimates by computing the accuracy of our part-level classifications. We report this accuracy as the average percentage of parts that are correctly classified.

We run 100 tests per scene and summarize our performance with MSD errors at the 50th, 75th, and 90th percentiles and our average assembly state accuracy.

1) Results: The performance of our proposed methods is summarized in Table II, where we report the MSD pose error and assembly state accuracy for three different approaches: the coarse method alone (Coarse), coarse followed by render-and-compare refinement (Coarse + RAC), and coarse followed by canonical voxel grid refinement (Coarse + Voxel). MSD is presented at the 50th, 75th, and 90th percentiles.

Fig. 6 provides examples of the estimated assembly states. Our analysis shows that the coarse method achieves reasonable accuracy for both pose and assembly state. The render-and-compare refinement method substantially reduces the pose error but offers marginal improvement in assembly accuracy. In contrast, the canonical voxel grid refinement method significantly enhances both pose and assembly accuracy.

TABLE II
ASSEMBLY RECOGNITION PERFORMANCE

	MSD (cm)			Assembly Accuracy (%)
	50th	75th	90th	
Engine				
Coarse	4.8	7.2	10.7	63.8
Coarse + RAC	2.5	4.5	7.9	65.0
Coarse + Voxel	3.6	5.7	8.8	70.6
Printer				
Coarse	3.5	4.9	7.4	78.7
Coarse + RAC	1.7	2.6	4.5	78.8
Coarse + Voxel	2.5	3.6	6.2	85.0

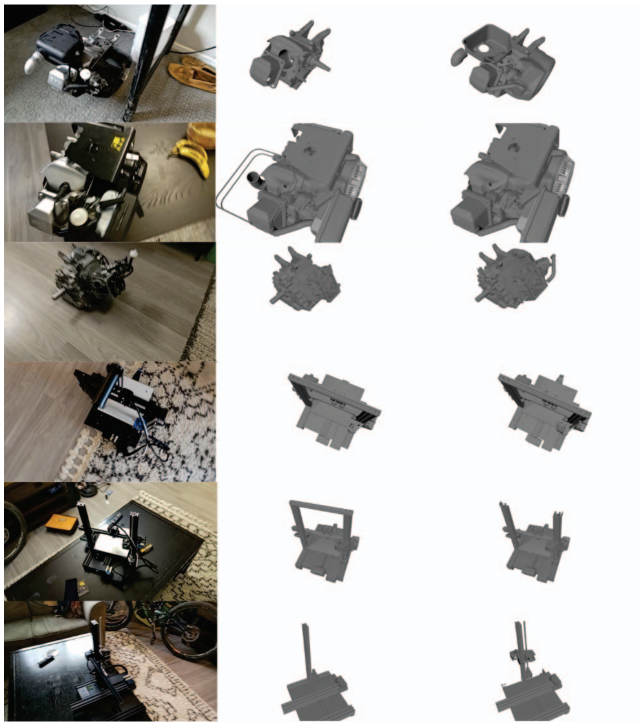

Fig. 6. Example assembly states estimates. Left column shows one of the six input images, while middle and right columns show the estimated pose and assembly state. The middle column is the coarse estimate and the right column is the refined estimate from the canonical voxel grid refinement.

2) Importance of Valid Assembly States: To highlight the importance of our concept of valid assembly states, we conducted an additional experiment where we trained the coarse network and the canonical voxel grid refinement network on synthetic data with random assembly states. In these states, the parts are still at their correct pose, but there may be gaps between parts, leading to physically implausible scenes (e.g. floating parts).

The results of this experiment, summarized in Table III, demonstrate a major decrease in performance compared to the results obtained when considering valid assembly states only (Table II).

TABLE III
PERFORMANCE WHEN TRAINED ON RANDOM ASSEMBLY STATES

	MSD (cm)			Assembly Accuracy (%)
	50th	75th	90th	
Engine				
Coarse	5.4	7.9	12.3	57.1
Coarse + Voxel	3.9	6.7	15.2	59.5
Printer				
Coarse	3.9	6.6	24.5	54.6
Coarse + Voxel	2.1	3.9	22.4	53.0

These results underscore the importance of the concept of valid assembly states in our approach. By taking into account the physical constraints and dependencies between the parts of the assembly, our method can achieve more accurate assembly state predictions and pose estimates.

E. Analysis

In our analysis, we focused on understanding the impact of various factors, including part observability and size, on the accuracy of our assembly classification.

We define observability based on whether we can directly see the presence or absence of a part. If a part is present and at least some of its surface is unoccluded in a view, it is observable. An absent part is observable if at least some of its surface would be unoccluded in a view, if it was present (i.e., it is clear that we are seeing behind the part). Unobservable is the opposite. So there are four possibilities for a part:

1) Absent and observable (AO)
2) Absent and unobservable (AU)
3) Present and observable (PO)
4) Present and unobservable (PU)

The results, shown in Table IV, revealed a notable trend: absent parts (both AO and AU) generally had higher accuracies compared to present parts (PO and PU). This may be attributed to the synthetic training data and the potential domain gap when applying the models to real-world scenarios. This gap could lead models to be more conservative, favoring the prediction of absence over presence.

As expected, we found that observable parts (AO and PO) consistently had higher accuracies than unobservable parts (AU and PU) across all categories, for both the engine and printer, and across all three methods (Coarse, Coarse + RAC, and Coarse + Voxel). This underscores the critical impact of observability on the model's ability to accurately classify assembly states.

In parallel, we conducted a linear regression analysis to understand the impact of part size on accuracy. We defined the characteristic length (”Length”) of a part as the diagonal of its 3D bounding box. The regression coefficients for Length, also presented in Table IV, indicate a strong positive effect on accuracy. Larger parts are generally recognized more accurately, aligning with the intuitive notion that bigger parts are easier to detect and classify. This effect was consistent across all methods and both test assemblies, albeit with varying degrees of influence.

TABLE IV
AVERAGE CLASSIFICATION ACCURACIES FOR OBSERVABILITY CATEGORIES (AO, AU, PO, PU) AND REGRESSION COEFFICIENT (β) FOR LENGTH.

	Average Accuracy (%)				Length
	AO	AU	PO	PU	Coef. (β)
Engine					
Coarse	91.2	90.1	46.1	32.7	1.53
Coarse + RAC	91.7	90.1	46.8	32.7	1.58
Coarse + Voxel	88.8	83.9	57.4	47.3	1.26
Printer					
Coarse	92.3	57.5	71.5	65.8	0.61
Coarse + RAC	92.6	55.9	71.7	65.5	0.64
Coarse + Voxel	90.3	52.0	82.6	80.0	0.34

The render-and-compare method (Coarse + RAC) showed minimal changes in classification accuracy compared to the coarse method alone, suggesting limitations in addressing the challenges of part observability and size. However, the canonical voxel grid refinement method (Coarse + Voxel) demonstrated a more robust performance, particularly in enhancing the recognition of present parts. This suggests that this method effectively integrates information from multiple views to improve accuracy, especially for smaller or less observable parts.

Our analysis highlights the complexities and nuances in assembly state recognition, emphasizing the importance of observability and part size. These insights not only validate the effectiveness of our methods but also point towards potential areas for future enhancement.

V. CONCLUSION

Our methods for assembly recognition have demonstrated significant potential in solving complex assembly tasks. The coarse method and refinement approaches have provided valuable insights into the problem of assembly recognition, specifically in the context of Augmented Reality guidance.

In particular, canonical voxel grid refinement is a novel contribution that has proven to be both accurate and potentially very efficient. Its utilization of a 3D voxel grid to integrate multi-view features offers a distinct advantage over traditional single-view techniques. Furthermore, if canonical voxel grid refinement shares 2D CNN encoder weights with the coarse method, it could offer even greater computational efficiency.

The results from our experiments highlight the impact of part size and occlusion on the accuracy of our system. These findings provide a valuable direction for future improvements. For example, higher-resolution images or more sophisticated feature extraction techniques could potentially address the issue of part size. Occlusion, however, may represent a more fundamental challenge, but our multi-view approach inherently mitigates some of its effects.

Our methods hold potential for wide-ranging applications, not only in AR-guided assembly tasks, but also in broader fields such as robotics and industrial automation. For instance, robots equipped with our system could potentially perform complex assembly tasks autonomously.

Looking forward, several avenues for future work emerge from our research. One promising direction involves exploring methods to further improve the realism of our synthetic training data. Another involves developing networks that can handle more complex assembly states, including those involving a larger number of parts or more complex interactions between parts.

REFERENCES

[1] M. Rad and V. Lepetit, “Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3848–3856.

[2] J. H. Rafael Radkowski and J. Oliver, “Augmented reality-based manual assembly support with visual features for different degrees of difficulty,” *International Journal of Human–Computer Interaction*, vol. 31, no. 5, pp. 337–349, 2015. [Online]. Available: https://doi.org/10.1080/10447318.2014.994194

[3] X. Wang, S. K. Ong, and A. Y. C. Nee, “A comprehensive survey of augmented reality assembly research,” *Advances in Manufacturing*, vol. 4, no. 1, pp. 1–22, Mar 2016. [Online]. Available: https://doi.org/10.1007/s40436-015-0131-4

[4] H. Liu, Y. Su, J. Rambach, A. Pagani, and D. Stricker, “Tga: Two-level group attention for assembly state detection,” in *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2020, pp. 258–263.

[5] J. Pang, J. Zhang, Y. Li, and W. Sun, “A marker-less assembly stage recognition method based on segmented projection contour,” *Advanced Engineering Informatics*, vol. 46, p. 101149, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034620301208

[6] Y. Hong, J. Zhang, H. Fan, Z. Lang, J. Pang, and Y. Hou, “A marker-less assembly stage recognition method based on corner feature,” *Advanced Engineering Informatics*, vol. 56, p. 101950, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034623000782

[7] T. Hodan, M. Sundermeyer, B. Drost, Y. Labbe, E. Brachmann, F. Michel, C. Rother, and J. Matas, “Bop challenge 2020 on 6d object localization,” arXiv:2009.07378, 2020. [Online]. Available: https://arxiv.org/abs/2009.07378

[8] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, “Blenderproc: Reducing the reality gap with photorealistic rendering,” 2019. [Online]. Available: https://arxiv.org/abs/1911.01911

[9] S. Zakharov, I. Shugurov, and S. Ilic, “Dpod: 6d pose object detector and refiner,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1941–1950.

[10] K. Park, T. Patten, and M. Vincze, “Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation,” in *The IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 7668–7677.

[11] G. Georgakis, S. Karanam, Z. Wu, and J. Kosecka, “Learning local rgb-to-cad correspondences for object pose estimation,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8966–8975.

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

[13] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic, “Cosypose: Consistent multi-view multi-object 6d pose estimation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[14] Z. Murez, T. van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich, “Atlas: End-to-end 3d scene reconstruction from posed images,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[15] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” http://pybullet.org, 2016–2021.

[16] N. Morrical, J. Tremblay, Y. Lin, S. Tyree, S. Birchfield, V. Pascucci, and I. Wald, “Nvisii: A scriptable tool for photorealistic image generation,” arXiv:2105.13962, 2021. [Online]. Available: https://arxiv.org/abs/2105.13962

[17] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[18] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[19] B. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *Journal of the Optical Society A*, vol. 4, pp. 629–642, 04 1987.

[20] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, p. 381–395, jun 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[21] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, “On the continuity of rotation representations in neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5745–5753.

[22] C. G. J. Petersen, “The yearly immigration of young plaice in the limfjord from the german sea,” *Rept. Danish Biol. Sta.*, vol. 6, pp. 1–48, 1896.

[23] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[24] P. Iakubovskii, “Segmentation models pytorch,” 2019. [Online]. Available: https://github.com/qubvel/segmentation_models.pytorch

[25] M. Matl, “Pyrender: Easy-to-use gltf 2.0-compliant opengl renderer for visualization of 3d scenes,” Available: https://github.com/mmatl/pyrender, 2019.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 06 2016, pp. 770–778.

[27] T. Hodaň, E. Brachmann, B. Drost, F. Michel, M. Sundermeyer, J. Matas, and C. Rother, “Results of the bop challenge 2019,” in *5th International Workshop on Recovering 6D Object Pose, ICCV 2019*, Seoul, Korea, 2019.