Final Report of AR-Guided Real-Time Spatial Tutoring System (ARTUS): Assembly King

Rongqian Chen

Abstract

This report presents the design, implementation, and evaluation of the AR-Guided Real-Time Spatial Tutoring System (ARTUS), developed for Meta Quest 3 and PC. ARTUS aims to facilitate intuitive and real-time interactions between coaches and trainees in application domains such as cooking, scientific experiments, and equipment repair. This document outlines the system's architecture, innovative features, challenges faced, and future possibilities.

1 Introduction

This system aims to help trainees do assembly work with real-time reference from the coach. For mechanical part assembly work, due to the massive amount of parts involves in the product and the complexity of the assembly work, people usually need to follow a manual instruction book. However, such traditional method require professional knowledge because they require users to understand the principles of how parts work together. The limitation of a manual interaction is the way it interacts with users: it shows the parts in 2D and fixed instructions for tutoring without knowing the current situations users might have. This might cause misunderstandings and mistakes in assembly works.

Thus, in our case, we design a real-time tutoring system. Enhanced with expert experience, Vision Language Model, and Computer Vision techniques, AssemblyKing is able to perform accurate and fast-response feedback for users. It can recognize the parts and understand assembly steps, based on user's current progress, it highlights the parts they need and provide tutoring instructions for users. User can follow coach assembly steps and learn the tasks in a relative short time.

2 System Design & Architecture

AssemblyKing is designed to facilitate real-time spatial tutoring through augmented reality (AR) for assembly training. The system supports both coach and trainee roles using Meta Quest Pro/3/3s headsets and a PC-based application. At its core, the system captures the coach's environment, processes the visual data, and delivers real-time guidance to trainees over a local area network (LAN). Key features include object detection, instruction generation, and seamless synchronization of processed video feeds.

The system architecture is shown in Figure 1. Due to the limitation of Quest 3 camera access, we use the casting function to capture and process the coach perspective. The coach uses the Cast function available on the Meta Quest 3 headset, which streams their view to the Oculus website. AssemblyKing accesses this video feed by running the `main.py` script on a PC, enabling real-time screen recording and object detection using a pre-trained model. The processed video feed, augmented with bounding boxes around detected objects, is synchronized with trainees via

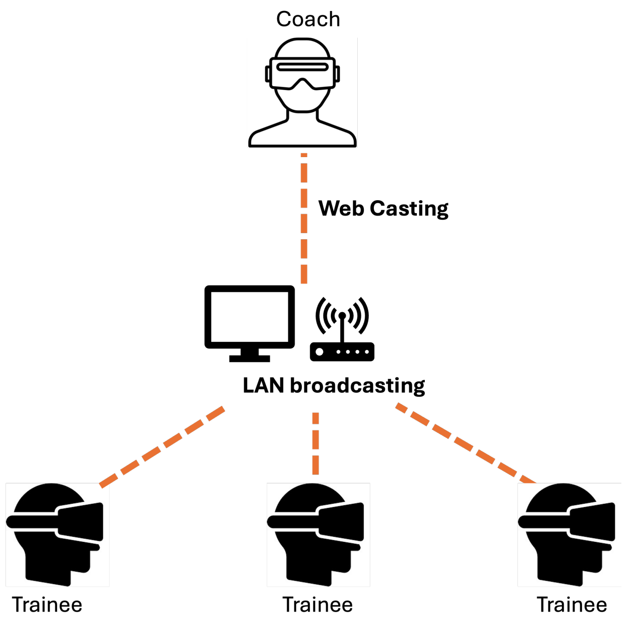


Figure 1: The system architecture of ARTUS.

a web server. The application generates a dynamic URL displayed in the terminal, which can be accessed on any browser connected to the same LAN. Figure 3 is an example of the website view when the trainee opens the website in the browser.

From the trainee's perspective, the system provides a user-friendly interface for viewing the coach's real-time augmented feed. Trainees connect to the system by entering the generated URL in their browser. The interface not only shows the coach's view with annotated bounding boxes for detected objects but also includes step-by-step instructions generated by a Vision-Language Model (VLM). These instructions dynamically adapt to the trainee's progress, providing a personalized and guided learning experience.

AssemblyKing is designed to scale effectively from one-to-many tutoring scenarios. The LAN-based synchronization ensures that multiple trainees can access the coach's feed simultaneously with minimal latency. The system dynamically adjusts to different network configurations by generating URLs specific to the local network, ensuring compatibility across various setups.

To achieve its functionality, AssemblyKing leverages several advanced technologies. Meta's Segment Anything Model 2 (SAM2) plays a critical role in dataset generation, providing precise object segmentation with minimal manual input and annotating the object automatically. The YOLOv8 model is used for real-time object detection after training on datasets generated using SAM2. To achieve a good performance, each object requires more than 300 images, with different distances and angles. In our case, we use the headset for video recording, the object is held by hand and manually change the distance and angles. A clear background is needed during the recording. Finally, we achieved 99.9% accuracy on the test set with 100 training epochs. Additionally, we use the Vision-Language Model (VLM) to generate intuitive instructions based on the current view, namely the ChatGPT-4o model. The input includes the object list, parts, and assembly instructions, enhancing the training experience for trainees. These technologies enable AssemblyKing to deliver a robust, scalable, and interactive AR-guided tutoring system. The details of the model are depicted in Figure 2.

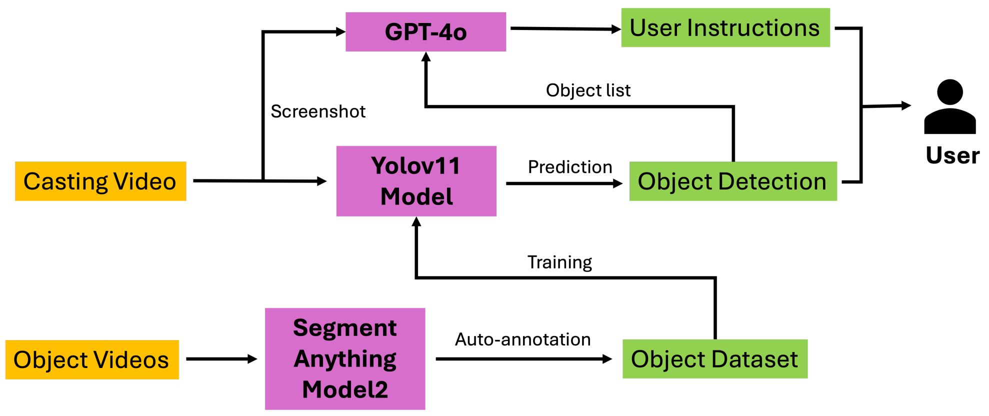


Figure 2: The model design of ARTUS.

3 Application Case Study

3.1 Domain Application

AssemblyKing is highly versatile and can be applied to any domain involving complex operations and the assembly of mechanical or non-mechanical parts. For this project, the selected domain focuses on mechanical assembly training, where trainees learn to assemble various components under the guidance of a coach. The system enables trainees to follow step-by-step instructions while visualizing object detection overlays, such as bounding boxes, on the components being assembled. This real-time guidance facilitates a hands-on learning experience, ensuring accuracy and reducing the risk of errors.

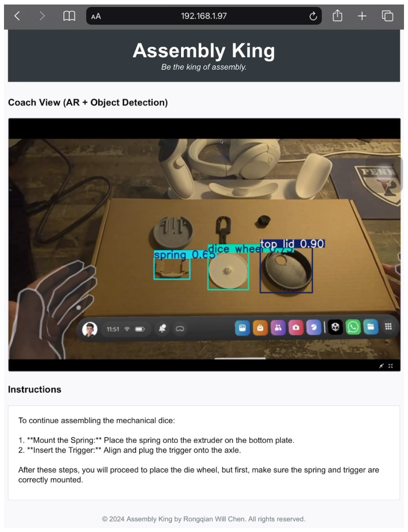
The coach's perspective is streamed in real-time using a Meta Quest 3 headset, while the system processes the video feed to detect objects and generate instructions for the trainee. This approach significantly improves the efficiency of training by making the assembly process more intuitive and interactive. For instance, during a mechanical assembly task, the system can highlight specific components and guide trainees on how to connect or align parts. Screenshots from the trainee perspective, as shown in Figure 3, demonstrate the bounding box annotations and instructional overlays that facilitate the learning process.

3.2 Versatility

AssemblyKing is designed to adapt to various contexts, making it suitable for a wide range of applications beyond mechanical assembly. For example:

- **Cooking**: The system can guide users through recipes, identifying ingredients and providing step-by-step instructions for preparation and cooking.
- **Scientific Experiments**: In a laboratory setting, the system can help trainees locate and identify tools or substances, ensuring proper handling and sequencing during experiments.
- **Equipment Repair**: The system can assist technicians in identifying faulty components and guiding them through the repair or replacement process.

The adaptability of the system lies in its ability to train on different datasets tailored to the specific domain. For instance, in the cooking application, the dataset might include images of

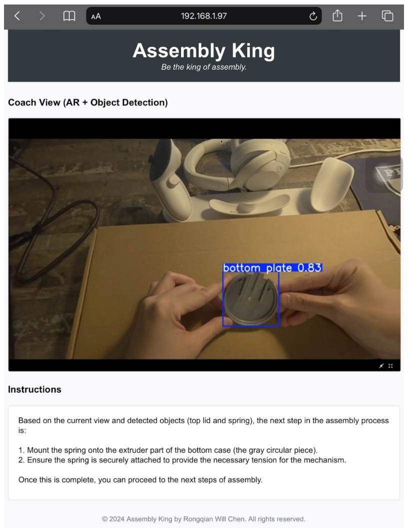


To continue assembling the mechanical dice:

1. **Mount the Spring:** Place the spring onto the extruder on the bottom plate.
2. **Insert the Trigger:** Align and plug the trigger onto the axle.

After these steps, you will proceed to place the die wheel, but first, make sure the spring and trigger are correctly mounted.

(a)

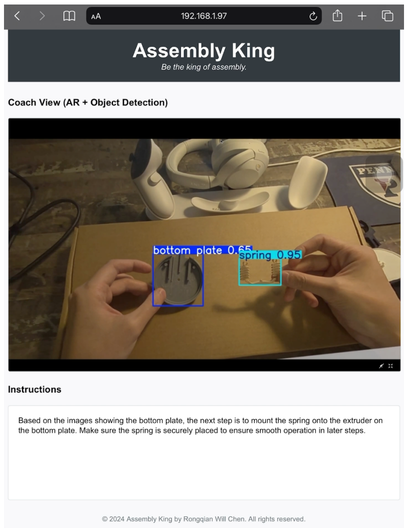


Based on the current view and detected objects (top lid and spring), the next step in the assembly process is:

1. Mount the spring onto the extruder part of the bottom case (the gray circular piece).
2. Ensure the spring is securely attached to provide the necessary tension for the mechanism.

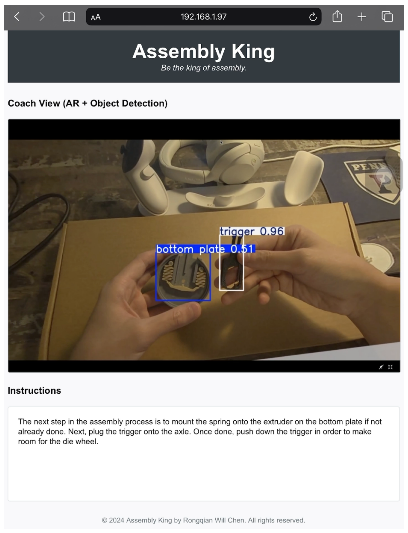
Once this is complete, you can proceed to the next steps of assembly.

(b)



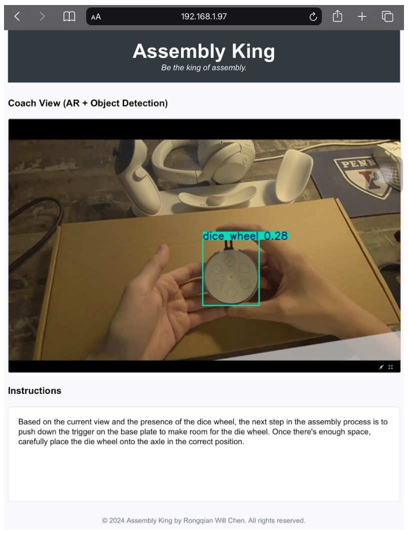
Based on the images showing the bottom plate, the next step is to mount the spring onto the extruder on the bottom plate. Make sure the spring is securely placed to ensure smooth operation in later steps.

(c)



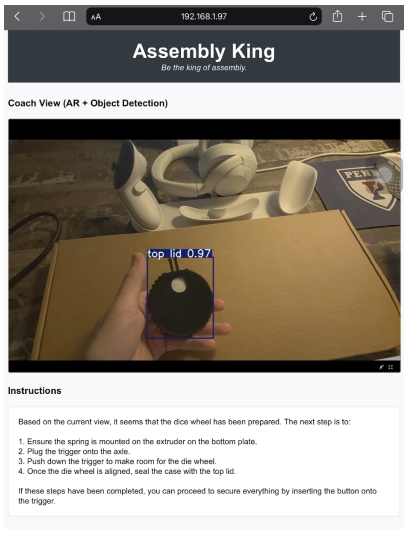
The next step in the assembly process is to mount the spring onto the extruder on the bottom plate if not already done. Next, plug the trigger onto the axle. Once done, push down the trigger in order to make room for the die wheel.

(d)



Based on the current view and the presence of the dice wheel, the next step in the assembly process is to push down the trigger on the base plate to make room for the die wheel. Once there's enough space, carefully place the die wheel onto the axle in the correct position.

(e)



Based on the current view, it seems that the dice wheel has been prepared. The next step is to:

1. Ensure the spring is mounted on the extruder on the bottom plate.
2. Plug the trigger onto the axle.
3. Push down the trigger to make room for the die wheel.
4. Once the die wheel is aligned, seal the case with the top lid.

If these steps have been completed, you can proceed to secure everything by inserting the button onto the trigger.

(f)

Figure 3: Screenshots from the trainee perspective, demonstrating various stages of interaction with the AR-Guided Tutoring System. The images show bounding box annotations and instructional overlays as seen by the trainee.

ingredients, utensils, and cooking steps. In contrast, for equipment repair, the dataset would focus on machine components and their interconnections.

As shown in the screenshots in Figure 3, the system effectively provides instructional overlays for different scenarios, ensuring a consistent user experience regardless of the application domain. This adaptability demonstrates the potential of AssemblyKing to become a valuable tool across diverse fields requiring training, assembly, or operational guidance.

4 Challenges & Lessons Learned

4.1 Technical Challenges

During the development, one of the biggest challenges is the integration of computer vision models and the Unity environment. Since the Unity requires C# for development, and there are rare resource of such models. Another option is develop sockets for python programs, but it due to the time constraint of this project, this option is not feasible. Finally, we choose to use python for image processing, segmentation and Vision-Language Model programming.

Another challenge is to access camera view of Quest3. Meta has limit the thrid party developer access the camera view due to privacy concerns. So the only way to get the view and process it at a external program is to use casting function.

4.2 Design Challenges

Developing AssemblyKing presented several design challenges, primarily revolving around ensuring real-time performance, system usability, and robust object detection. One major issue was the latency introduced during video processing and synchronization between the coach and trainee. To address this, the system was optimized by reducing the size of transmitted data and fine-tuning the server architecture to prioritize low-latency operations.

Another challenge involved creating a user-friendly interface for both the coach and the trainee. On the coach's side, it was essential to ensure the system's ability to seamlessly capture and process the video feed without disrupting the natural workflow. For the trainee, the bounding box annotations and instructional overlays needed to be intuitive and clearly visible, even in varying lighting conditions or when objects were partially occluded. These challenges were resolved by thorough testing in diverse environments and implementing adaptable rendering techniques.

Lastly, the training dataset posed a challenge. The system relies heavily on the accuracy of the YOLOv8 model, which in turn depends on high-quality annotated data. The initial manual annotation of object boundaries using SAM2 was time-consuming. This issue was mitigated by creating semi-automated pipelines for dataset generation, which significantly reduced the effort required for dataset preparation while maintaining accuracy.

4.3 Lessons Learned

The project provided valuable insights into developing real-time AR systems. One key lesson was the importance of optimizing data pipelines to minimize latency, especially in scenarios requiring real-time interactions. It also highlighted the necessity of user-centered design when creating interfaces for both coaches and trainees. Feedback from users during testing helped refine the system, ensuring that it met their practical needs.

Another significant takeaway was the critical role of robust dataset preparation in training accurate object detection models. By leveraging tools like SAM2 and improving the dataset generation process, we learned how to achieve high-quality annotations efficiently. Additionally, integrating a Vision-Language Model (VLM) demonstrated the potential of AI-driven instructions in enhancing user guidance, reinforcing the importance of leveraging state-of-the-art technologies to create a seamless learning experience.

5 Conclusion & Future Work

5.1 Achievements

AssemblyKing successfully achieved its primary objectives of providing real-time AR-guided spatial tutoring for assembly tasks. The system demonstrated effective synchronization between the coach and trainee, enabling intuitive interactions through object detection and instructional overlays. By leveraging cutting-edge technologies like Meta's SAM2, YOLOv8, and a Vision-Language Model, the system offered a robust, scalable, and adaptable solution for training in assembly and other application domains.

The system also excelled in its adaptability to various contexts, showcasing its versatility in domains such as mechanical assembly, cooking, and equipment repair. The use of AR significantly enhanced the learning experience by providing visual guidance, reducing errors, and improving the trainee's understanding of tasks. The scalability to 1-to-n tutoring further solidified its potential as a training platform.

5.2 Future Enhancements

While AssemblyKing achieved its core goals, several areas for improvement and extension remain. One potential enhancement involves expanding the system's scalability to include remote training over the internet, allowing trainers and trainees to connect from different locations. This would require further optimization of the video processing pipeline to handle higher latencies and varying network conditions.

Another improvement could involve integrating more advanced computer vision techniques, such as 3D object detection and pose estimation, to provide even more precise guidance during complex tasks. Additionally, the Vision-Language Model could be expanded to offer contextualized instructions that adapt dynamically to the trainee's progress and errors in real-time.

Finally, creating a more extensive and diverse dataset would enhance the system's adaptability to new domains and objects. Incorporating user feedback into future iterations would ensure continued improvements in usability and functionality. These enhancements would position AssemblyKing as a leading tool for AR-guided training across a wide range of industries and applications.

References