Progress Observation in Augmented Reality Assembly Tutorials Using Dynamic Hand Gesture Recognition

Tania Kaimel*[1] Ana Stanescu†[1] Peter Mohr[1] Dieter Schmalstieg [1,2] Denis Kalkofen [1,3]

[1]Graz University of Technology [2]University of Stuttgart [3]Flinders University

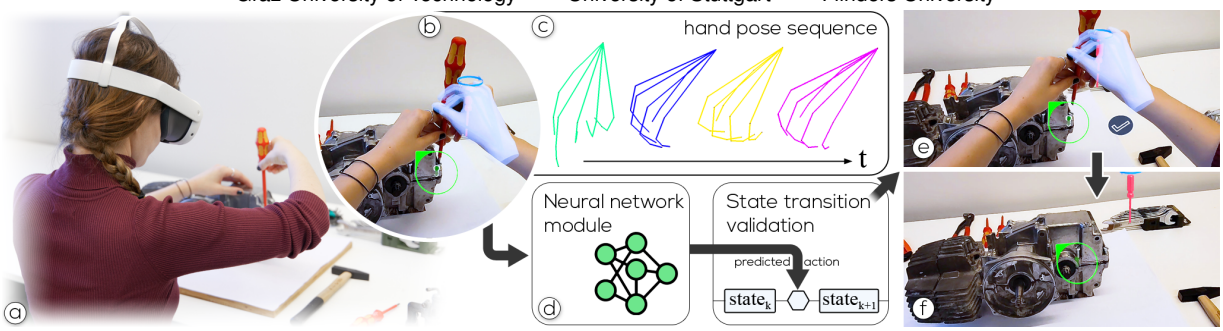

Figure 1: Augmented Reality assembly tutorials. (a) The user follows visual instructions within the 3D workspace. In this example, the user removes the highlighted object part using a screwdriver. (b) During interaction, hand poses (c) are classified using a neural network (d) to observe the state of the system (e) and detect when to proceed automatically to the next step (f).

Abstract

We propose a proof-of-concept augmented reality assembly tutorial application that uses a video-see-through headset to guide the user through assembly instruction steps. It is solely controlled by observing the user's physical interactions with the workpiece. The tutorial progresses automatically, making use of hand gesture classification to estimate the progression to the next instruction. For dynamic hand gesture classification, we integrate a neural network module to classify the user's hand movement in real time. We evaluate the learned model used in our application to provide insights into the performance of implicit gestural interactions.

1 Introduction and Related Work

Assembly tutorials commonly use a step-by-step design consisting of a sequence of steps which are presented one after another. Efficiently following step-by-step tutorials requires synchronizing the presentation with the progress of the user [5]. In Augmented Reality (AR), assembly tutorials have mostly relied on manual step progression [9], and only recently used detection of the current object configuration from 2D or 3D scene observations [7, 8]. However, poor image data, small object parts, and heavy object occlusions during user interaction often make it difficult to confidently detect the object configuration at run-time. Thus, in this work, we focus on using the hand motions of the user who follows an AR assembly tutorial to detect the completion of a step.

Several research projects have investigated the applicability of gesture recognition for AR tutorials before. For example, the very recent work of Reza et al. [6] focuses on detecting hand gestures for interactions in AR. This approach is similar to ours in its classification of dynamic hand gestures for an AR application. In contrast to our work, it deals with interactions with individual measuring tools and object parts, rather than with object assembly, where a correct sequence of assembly states must be verified.

*e-mail: tania.kaimel@student.tugraz.at

†e-mail: stanescu@tugraz.at

Coupetè et al. [1] propose using gesture recognition to support human-robot interaction in assembly tasks. An RGB-D camera records the scene from a top view and tracks the user's hands. The extracted data is used to train a hidden Markov model. The work of De Smedt et al. [2, 3] introduces a hand gesture classification approach based on support vector machines as well as the DHG dataset, which contains hand skeleton poses with different gestures, recorded from a third person view. We focus on a more concrete use case with custom gestures, including the presence of tools.

The HIGS system of Lu et al. [4] automatically extracts video instructions, including hand gestures from user demonstrations, which are subsequently used to guide new users through the tasks. The guidance is based on extracting hand joints from frames and computing the proximity to objects. In contrast to our work, this approach uses a monitor to show localized video snippets and relies on Euclidean distances to detect which gesture has been performed.

2 Method

Our AR application implements a step-by-step assembly tutorial that enables a hands-free transition between instruction steps by classifying the user's action. We classify the gestures in real time using a neural network. The application is implemented in Unity. The object used in our experiments is an engine (Figure 1), consisting of three main parts, four screws, and two nuts that need to be fastened. Spare parts are located in a bin on the right of the work area.

AR application Direct application control is only required for starting and stopping the application and during registration. For object registration, we build on Meta's *Scene Model*[1] for initial object placement. It enables the user to define areas such as desks or chairs and use them as permanent scene anchors that can be re-localized after an application restart. To offer a semi-automatic registration of the initial object part, we ask the user to identify the working area with the hand controllers upon first use. The initial object is placed in the center of the work area and the user then is instructed to make small pose adjustments with their hand to ensure proper object registration.

[1]https://www.meta.com/de-de/blog/quest/mixed-reality-definition-passthrough-scene-understanding-spatial-anchors/

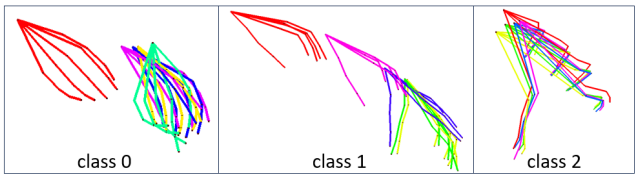

Figure 2: Example hand sequences from all used classes: rotation (0), removal (1), and addition (2).

We show instructions by highlighting registered 3D parts of the object. The next step is indicated by overlaying a semi-transparent green model over the part that needs to be handled. In case a tool is needed, a virtual copy of it is additionally displayed on the side. During the tutorial, the current instruction is shown along with icons that provide feedback on hand movements, as classified by the neural network. If the predicted class of the current gesture coincides with the expected class, according to the assembly sequence of the object, a checkmark icon appears next to the highlighted object (Figure 1(e)). A wrong gesture prompts the display of an "X" icon. After the last instruction step is completed, the application visually confirms a successful assembly.

Classification The classification runs on a desktop PC and communicates with the Unity application using UDP over WiFi. The module uses the MMskeleton framework [11] with the ST-GCN network [10]. We adapted the input format from the whole body skeleton to the hand skeleton, in a format specifying connections from the wrist (defined as root) to the individual fingers, leading to a total of 12 joints. The classes for hand actions are rotation, removal (pick-up), and addition of object parts, as shown in Figure 2.

Since our gestures are dynamic, we must choose a time window in which to capture hand movement. We empirically determined a duration of 3 s, the approximate time needed for the gestures used. For example, a removal action rarely takes longer than 3 s. If the recordings are longer, the sequence would contain irrelevant data for this gesture, while shortening the interval would cut off the gesture. We extract 10 samples of 0.3 s length from such a 3 s recording.

Data Recordings Our application offers a recording functionality where sequences of hand poses can be saved as datasets. For automatic class labeling, we make use of a bounding-box-like system that uses thresholds based on the size of the virtual object to determine if the user's hand is in reach of the object, indicating that the gesture has started. In order to decide if a hand is in reach, the positions of the wrist, the thumb and the index finger are considered. As soon as this requirement is met, data recording starts. To start recording the next instruction, the user's hands had to cycle once through in-reach and out-of-reach states.

Data recording was carried out by an expert. We converted the 3D joint positions into the coordinate system of the hand and normalized with regard to the first wrist position of each sample. We applied data augmentation by adding small random offsets to each joint's coordinates. Our dataset contains 1038 samples in the training set and 475 samples in the validation set, a total of 1513 samples.

3 Results and Future Work

To explore which combination of hand joints performs best, we evaluated several variants of the dataset, either with the entire hand or by leaving out some fingers. We aim to investigate whether occlusions of parts of the hand have a negative impact on the results.

We tested the system with batch sizes 8 and 16, and with different numbers of epochs ranging from 80 to 400. Best results were achieved using a batch size of 8, 250 epochs for training and a learning rate of 0.0001. We run the experiments on an NVIDIA GeForce 3080 Ti graphics card and display AR on a Meta Quest 3. We train using a three-fold cross-validation. The results obtained by testing on validation set are averaged for the folds (Table 1). The best results are obtained on the whole hand (Figure 3), but high accuracy, precision and recall are achieved in all test configurations with real-time inference.

Dataset	Top1 Acc.	Recall	Precision
All hand joints	95.80	95.55	**95.15**
Thumb+index +wrist	**95.88**	**95.87**	93.31
Without index	94.50	94.17	94.33

Table 1: Classification results with different combinations of joints.

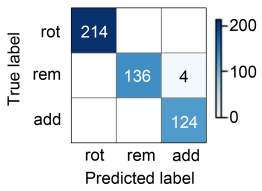

Figure 3: Confusion matrix of the best performing model (all hand joints).

As a proof of concept, our work has clear limitations. If similar gestures are needed at a certain point in the assembly for different possible steps, the application may not be able to distinguish them. A further evaluation of gesture detection could offer more insight into the robustness and applicability of our work. Recording more data or combining our data with a dataset such as DHG [3] could provide more general results.

In conclusion, we demonstrate how the natural use of hands in assembly can facilitate automatic progress observation in AR assembly applications. The noteworthy performance of the skeleton action recognition indicates how this approach can expand the vocabulary of spatial interaction techniques for AR applications targeting everyday use.

Acknowledgments

Dieter Schmalstieg is supported by the Alexander von Humboldt Foundation and the German Federal Ministry of Education and Research. The authors wish to thank Professor Rudolf Pichler from the smartfactory@tugraz lab for the shared data recordings.

References

[1] E. Coupeté, F. Moutarde, and S. Manitsaris. Gesture recognition using a depth camera for human robot collaboration on assembly line. *Procedia Manufacturing*, 2015.

[2] Q. De Smedt, H. Wannous, and J.-P. Vandeborre. Skeleton-based dynamic hand gesture recognition. In *Proc. CVPRW*, pp. 1–9, 2016.

[3] Q. De Smedt, H. Wannous, J.-P. Vandeborre, J. Guerry, B. L. Saux, and D. Filliat. 3d hand gesture recognition using a depth and skeletal dataset: Shrec'17 track. In *Workshop on 3D Object Retrieval*, pp. 33–38, 2017.

[4] Y. Lu and W. Mayol-Cuevas. Higs: Hand interaction guidance system. In *IEEE ISMAR Adjunct*, 2019.

[5] S. Pongnumkul, M. Dontcheva, W. Li, J. Wang, L. Bourdev, S. Avidan, and M. F. Cohen. Pause-and-play: automatically linking screencast video tutorials with applications. In *ACM UIST*, pp. 135–144, 2011.

[6] S. Reza, Y. Zhang, O. Camps, and M. Moghaddam. Towards seamless egocentric hand action recognition in mixed reality. In *IEEE ISMAR-Adjunct*, pp. 411–416, 2023.

[7] A. Stanescu, P. Mohr, M. Kozinski, S. Mori, D. Schmalstieg, and D. Kalkofen. State-aware configuration detection for augmented reality step-by-step tutorials. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 157–166, 2023.

[8] A. Stanescu, P. Mohr, D. Schmalstieg, and D. Kalkofen. Model-free authoring by demonstration of assembly instructions in augmented reality. *IEEE Transactions on Visualization and Computer Graphics (special issue ISMAR)*, 28(11):3821–3831, 2022.

[9] M. Yamaguchi, S. Mori, P. Mohr, M. Tatzgern, A. Stanescu, H. Saito, and D. Kalkofen. Video-annotated augmented reality assembly tutorials. UIST '20, p. 1010–1022, 2020.

[10] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[11] S. Yan, Y. Xiong, J. Wang, and D. Lin. Mmskeleton: open source toolbox for skeleton-based human understanding. `https://github.com/open-mmlab/mmskeleton`, 2019.