Learning Object-Action Relations from Bimanual Human Demonstration Using Graph Networks

Christian R. G. Dreher, Mirko Wächter, and Tamim Asfour

***Abstract*—Recognizing human actions is a vital task for a humanoid robot, especially in domains like programming by demonstration. Previous approaches on action recognition primarily focused on the overall prevalent action being executed, but we argue that bimanual human motion cannot always be described sufficiently with a single action label. We present a system for frame-wise action classification and segmentation in bimanual human demonstrations. The system extracts symbolic spatial object relations from raw RGB-D video data captured from the robot's point of view in order to build graph-based scene representations. To learn object-action relations, a graph network classifier is trained using these representations together with ground truth action labels to predict the action executed by each hand. We evaluated the proposed classifier on a new RGB-D video dataset showing daily action sequences focusing on bimanual manipulation actions. It consists of 6 subjects performing 9 tasks with 10 repetitions each, which leads to 540 video recordings with 2 hours and 18 minutes total playtime and per-hand ground truth action labels for each frame. We show that the classifier is able to reliably identify (action classification macro F_1-score of 0.86) the true executed action of each hand within its top 3 predictions on a frame-by-frame basis without prior temporal action segmentation.**

***Index Terms*—Learning from demonstration, semantic scene understanding, visual learning.**

I. Introduction

FOR domains like programming by demonstration [1], it is vital for a robot to be able to recognize the actions of a human. The obtained information can be used to learn action sequences from a human teacher in order to replicate tasks, or anticipate what a human wants to do to timely assist them. Most previous approaches (e.g., [2]–[10]) on action recognition usually assigned one single action label to each point in time, but we argue that this is not enough in general, considering natural bimanual human motion. Take, for example, a baking task where one has to fold egg whites into a dough. This implies two actions, as one is required to *pour* egg whites into a bowl with one hand, while *folding* them in with the other. This simple example is not easily representable with a single label, but assigning an action label to each hand solves this. Especially bimanual programming by demonstration approaches [11] could benefit from this granularity of information in order to individually discriminate the semantic role of each hand.

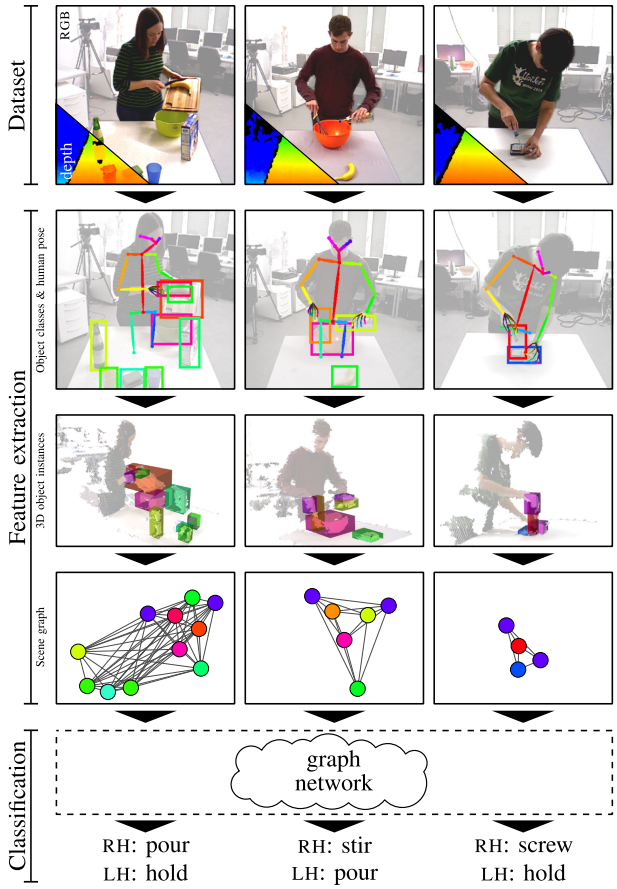

Fig. 1. Simplified overview and outline of our contributions in context. *Dataset*: 3 exemplary RGB-D images from the dataset (images edited for better clarity); *Feature extraction*: The interim results and final result of the 3-stage processing pipeline, a scene graph; *Classification*: A graph network classifier making predictions about the performed action for the right hand (RH) and the left hand (LH) based on the scene graph.

One important aspect of programming by demonstration is the question of how to form an abstract knowledge representation. Using symbolic features for that has several benefits, as raw video data streams are of high dimensionality and have no inherent semantic meaning. Additionally, this provides the desired abstraction layer, both for the representation, as well as for the elicitation of the symbolic features. An example for this representation is a robot observing a human teacher who pours water from a bottle into a cup. It is more general to

Manuscript received June 7, 2019; accepted October 9, 2019. Date of publication October 23, 2019; date of current version November 29, 2019. This letter was recommended for publication by Associate Editor P. Falco and Editor D. Lee upon evaluation of the reviewers' comments. The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement Number 643950 (SecondHands) and the Carl Zeiss Foundation. *(Corresponding author: Christian R. G. Dreher.)*

The authors are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe 76131, Germany (e-mail: c.dreher@kit.edu; mirko.waechter@kit.edu; asfour@kit.edu).

Digital Object Identifier 10.1109/LRA.2019.2949221

memorize the scene in a symbolic way (i.e., while *pouring*, the *bottle* is *above* the *cup*) instead of determining exact coordinates of the corresponding objects. In this work, we focus on 3D symbolic spatial relations between the human hands and the objects for each given point in time, and represent this scene as a graph. Object-action relations are learned by training a graph network [12] classifier, a machine learning building block, with those scene graphs. In particular, the classifier learns to estimate action labels for each hand given a history of scene graphs. In order to evaluate, which spatial relations between a given pair of objects and the human hands are in effect, RGB-D video data was used.

To conclude this introduction, the main contributions of our work (presented in Section III) are as follows:

- A novel RGB-D video dataset specifically tailored to research bimanual human actions.
- A pipeline to construct scene graphs from RGB-D videos.
- A frame-wise action segmentation and recognition approach, which is invariant to the number or order of object instances, does not require a prior temporal segmentation, and predicts actions for each hand individually by learning object-action relations.

Fig. 1 shows the contributions in context. The whole RGB-D dataset, as well as supplementary and derived data, are publicly available at bimanual-actions.humanoids.kit.edu.

II. Related Work

In this section, multiple related works are considered regarding their datasets, and their methods towards predicting human actions from various input modalities.

A. Datasets

There are several video datasets compiled to research human action recognition problems, where the recording modalities range from RGB only [13]–[15] over RGB-D [16]–[21] to complex multimodal environments [22].

Kühne *et al.* [13] presented a large RGB dataset of cooking activities, called the *Breakfast Actions Dataset*, with a total length of over 77 h. Damen *et al.* [14] compiled the EPIC KITCHENS dataset, where subjects were asked to wear a head-mounted GoPro to record their cooking activities. With vast amounts of videos being publicly available on the internet, Zhou *et al.* [15] collected 176 h of instructional videos for their YouCook2 dataset. The selected videos were temporally segmented into procedure steps, but mostly contain cuts. Since we wanted to evaluate 3D spatial relations between objects, we could not make use of any of these datasets. Apart from that, they were not suitable for our scenario where a robot observes a human teacher, because they either do not provide a viewing angle as seen from the robot, or are not continuous in time.

Wu *et al.* [16] compiled a large RGB-D dataset for action recognition. It features 7 subjects in 458 high quality recordings and has a total length of about 3 h 50 min. Other RGB-D datasets from the Cornell University are the CAD-60 with Sung *et al.* [17], and the CAD-120 with Koppula *et al.* [20]. The datasets differ in the granularity of the annotation, and in size, as the CAD-120 is twice as large. In both datasets, the camera angle relative to the subject varies. Wang *et al.* [18] collected a dataset of activities of daily living, recorded from a fixed RGB-D camera in front of a sofa. Xia *et al.* [19] had 10 subjects perform 10 indoor activities in front of a fixed camera. Aksoy *et al.* [21] introduced the MANIAC dataset, which features 5 subjects over 140 RGB-D videos in total. Most of these datasets were recorded using a Microsoft Kinect. All of them, however, did not suit our needs, mostly because of the viewing angles, their small number of recordings, or their focus on activities rather than on fine-grained actions.

There are also other approaches, like the TUM Kitchen dataset, collected by Tenorth *et al.* [22]. It was recorded in an intelligent kitchen with several RGB cameras mounted on the ceiling and other kinds of sensors, and they considered the left hand and right hand separately for the ground truth. But again, our focus lies in the sensors available on a humanoid robot in a one-on-one scenario. With the exception of this dataset, none of the others discussed here, regardless of the modalities, considered bimanual actions, but instead focused on the overall prevalent activity or action.

For more extensive comparisons we refer to Poppe [23], Weinland *et al.* [24], or Chaquet *et al.* [25], where most of these datasets were discussed in great detail. Additionally, Zhang *et al.* [26] specifically surveyed RGB-D datasets for action recognition.

B. Action Recognition

Similar to the datasets, also the action recognition approaches can be divided into those who use RGB-D data [2], [3], RGB data only [4]–[7], and others [8]–[10], [22].

In many cases, conditional random fields (CRFs) were employed, a probabilistic graphical machine learning approach [2], [5], [22]. Koppula and Saxena [2] used a CRF to classify action segments, and therefore heavily relied on a prior accurate temporal segmentation. Kjellström *et al.* [5] used a CRF to learn object-action relations. Their method simultaneously classified and segmented actions, but only considered one hand. Tenorth *et al.* [22] considered both hands, but evaluated only on the left hand. All of these approaches model their problem in a chained graph structure, which is required so that the inference on CRFs is feasible.

Some early approaches interpret an action as a spatio-temporal volume of image frames over time, extracting the shape of the action by subtracting the background [6], [7]. In a more modern interpretation of this approach, Ji *et al.* [4] used a 3D convolutional neural network to not only convolve spatially, but also temporally.

Aksoy *et al.* [8] coined the term of *Semantic Event Chains* (SECs), a concept which encodes transitions between object relations in a matrix. The work on SECs was further continued by Ziaeetabar *et al.* [3], where SECs were enriched with a large array of static and dynamic spatial relations. In order to evaluate the spatial relations, 3D bounding boxes estimating the objects were used, calculated from RGB-D images.

Wächter and Asfour [9], as well as Mandery *et al.* [10], used the change of contact relations as strong indicator for the presence of temporal segmentation boundaries.

Action recognition is a common problem, and therefore there are a multitude of other approaches available. For a broader overview on older methods not mentioned here, we again refer to the works of Poppe [23] or Weinland *et al.* [24]. More recent approaches focusing on RGB input data are discussed by Herath *et al.* [27].

Except for that of Tenorth *et al.* [22], none of these works consider natural bimanual actions in the sense that each hand

Fig. 2. Exemplary recordings from our proposed dataset. First row: Preparing breakfast cereals by cutting and pouring a banana into a bowl, followed by milk and cereals. Second row: Cooking by stirring in a bowl while pouring water from a bottle to it. Third row: Disassembling a hard drive by unscrewing and removing a screw. (Images cropped/edited and depth images omitted for the sake of clarity and brevity.)

may perform an individual action, like stirring in a bowl while pouring water in it. Additionally, the used machine learning approaches often limit the application to a fixed set of object instances and to a specific order. We are only bound to a fixed set of object *classes*, multiple instances can easily be represented in the scene graph. Other than that, we do not require any prior temporal segmentation.

III. Approach

In this section, we will present our proposed approach for the segmentation and recognition of bimanual actions by learning the relation between objects and actions. The 3 contributions, also depicted in Fig. 1, are described in detail in the following subsections. First, we describe the RGB-D dataset collected to train the developed classifier in Subsection III-A. In Subsection III-B, we present a feature extraction pipeline, which takes such RGB-D data as input and constructs a scene graph. The nodes in such a scene graph encode the object classes (including hands), and an edge encodes the relations between two objects. The objects are detected using an object detection framework, while the hands are detected using a human pose estimation framework, both taking RGB images as input. Finally, in Subsection III-C, we introduce the main contribution of this paper. To learn object-action relations, i.e., the relation between a scene graph and the executed action, we employ a graph network classifier, a type of machine learning building block designed to operate on variable-sized graphs.

A. Dataset

For this work, a rich RGB-D video dataset of bimanual action sequences was compiled, 3 of which are shown exemplarily in Fig. 2 in a few key frames. We recorded 6 subjects (3 female, 3 male; 5 right-handed, 1 left-handed) performing 9 different tasks (5 in a kitchen context, 4 in a workshop context) from a robot's point of view. The considered tasks were 1. and 2. *cooking* in two variants (pour from bottle vs. pour from bowl), 3. *pouring water*, 4. *wiping the table*, and 5. *preparing breakfast cereals* for the kitchen tasks, as well as 6. and 7. *disassembling a hard drive* in two variants (hard drive on the table vs. in the hand), 8. *hammering nails*, and 9. *sawing wood* for the workshop tasks. Each task was repeated 10 times. This totals to 540 recordings of fully labeled bimanual actions with a total runtime of approx. 2 h 18 min. More precisely, one annotator manually labeled the whole dataset frame-wise once for each hand with one of 14 possible action classes in $A = \{$*idle*, *approach*, *retreat*, *lift*, *place*, *hold*, *stir*, *pour*, *cut*, *drink*, *wipe*, *hammer*, *saw*, *screw*$\}$. Wächter and Asfour [9] used a similar detail of labeling in which the hand *approaches* an object and, after using it, *retreats*. In most cases, the object also has to be *lifted* and *placed* for usage (e.g., *pouring*).

Apart from the objects the subject interacted with, up to 3 additional known and contextually fitting objects were placed on the table. The 12 considered object classes are $O = \{$*cup*, *bowl*, *whisk*, *bottle*, *banana*, *cutting board*, *knife*, *sponge*, *hammer*, *saw*, *wood*, *screwdriver*$\}$. For the classes *cup*, *bottle*, *bowl*, and *sponge*, several differently looking objects were used, as can be seen in Fig. 2. Additionally, 5413 frames (about 10 random frames per recording) were manually labeled with object class bounding boxes by the same annotator. This data was later used to train an object detection framework.

The hardware used to record the dataset was a PrimeSense Carmine 1.09 RGB-D camera, which captures images at 30 fps with a resolution of 640 px $\times$ 480 px. It was attached to a tripod at a height of 1.7 m to simulate a standing robot. The camera was tilted forth, so that the entire working space and the teacher's head were still pictured. Due to technical limitations at the time of recording the first subject, 83 of the 540 recordings had to be captured at only 15 fps.

B. Feature Extraction

The feature extraction is a vital link in our work to convey RGB-D images to scene graphs, on which our classifier operates on. We chose to work with symbolic features as opposed to following an end-to-end approach, since this greatly reduces the problem dimensionality. This means that less data is required, but also that these symbolic features need to be extracted first. To do so, a 3-stage pipeline was deployed. It was implemented

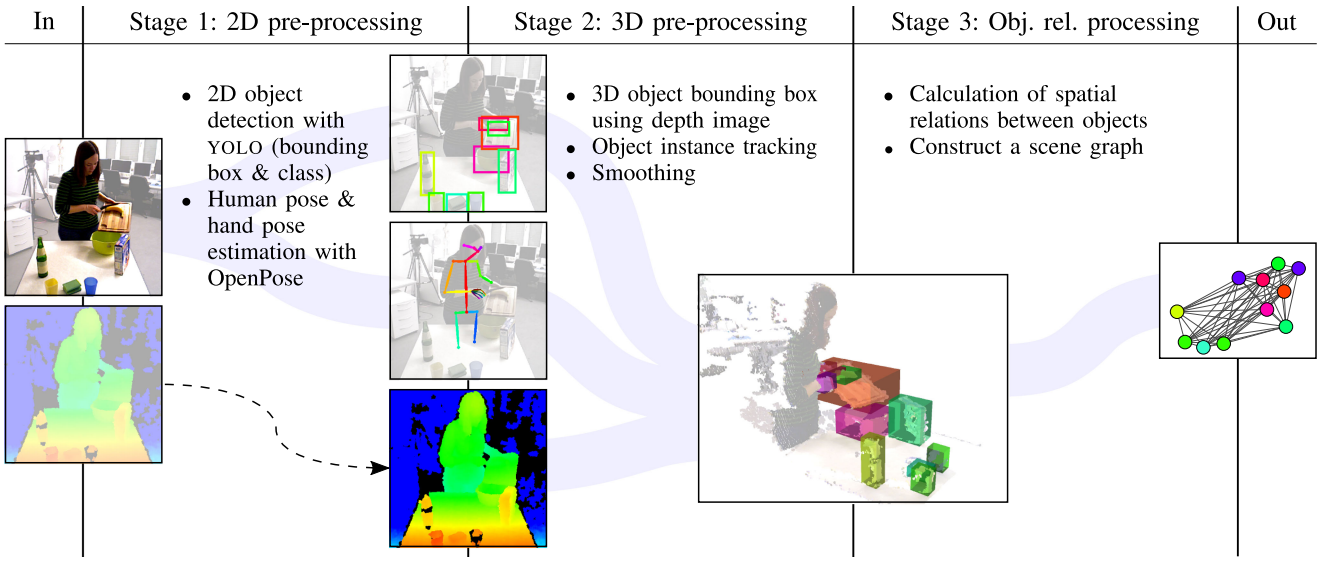

Fig. 3. Schematic of the 3-stage processing pipeline. Input: An RGB-D image. Output: A scene graph. Stage 1: 2D pre-processing, computing the 2D object bounding boxes and the human pose from the RGB image, forwarding the depth image to stage 2. Stage 2: 3D pre-processing, computing 3D object bounding boxes, tracking object instances, and smoothing the noise resulting from the depth image. Stage 3: Object relation processing, evaluating which spatial relations between each pair of objects are in effect, constructing a scene graph. For each stage, the inputs are depicted on the left border and the outputs on the right.

in C$_{++}$ and integrated into the ArmarX framework [28]. Fig. 3 shows a schematic of the whole processing pipeline. To give a brief overview: The input of the pipeline are consecutive RGB-D images, however only one RGB-D image is fed into the pipeline per pass. While the RGB image is used in the first stage, the depth image is not needed until the second stage. The output of the pipeline is a scene graph, where all detected object instances are represented as nodes, and all present relations between each pair of objects are encoded into the edges. Each pass of the pipeline takes about 250 ms with an Nvidia TITAN X. The outputs of all stages were recorded for every frame and are available for download as well. The following paragraphs will describe each stage in more detail.

The first stage in the pipeline is the 2D pre-processing, where the objects are detected with YOLO [29] (trained on the objects in our dataset) and the hands of the human teacher with OpenPose [30] by feeding the RGB image. The hand key points provided by OpenPose are used to calculate the 2D bounding box for each hand. Hence, this stage outputs a list of 2D bounding boxes of the objects detected by YOLO and the hands detected by OpenPose. Note that the hands are treated as any object in the following.

The second stage performs the 3D pre-processing, where the data of the first stage is used in conjunction with a point cloud derived from the depth image to acquire the 3D bounding boxes of the objects. This is achieved by clustering only that part of the point cloud, which is outlined by the 2D bounding boxes, and the assumption, that the biggest cluster in terms of point count belongs to the detected object. Minimum and maximum extents of that cluster yield the 3D bounding box. Since the depth images suffer from high-frequency noise, which directly transfers to the 3D bounding boxes, this stage also performs a smoothing by applying a Gaussian filter over the parameters of the past observed 3D bounding boxes of each object. The Gaussian filter was parameterized so that $3\sigma = 250\,\text{ms}$. Furthermore, to be able to apply the smoothing and to later calculate dynamic spatial relations between objects, it is important to identify concrete object instances over several frames. This stage therefore also includes an object tracking algorithm. The output of this stage are 3D object bounding boxes enriched by globally unique object instance identifiers.

The third stage is the object relation processing. The 3D bounding boxes of the previous stage are used to determine, which of the spatial relations are present for a given pair of objects. We considered the 15 spatial relations from Ziaeetabar *et al.* [3], namely $R = \{$ *contact*, *above*, *below*, *left*, *right*, *front*, *behind*, *inside*, *surround*, *moving together*, *halting together*, *fixed moving together*, *getting close*, *moving apart*, *stable* $\}$. Contrary to their formulation, however, no exception conditions were used. The output of this stage, and therefore the pipeline, is a scene graph, where nodes represent object instances, and edges encode spatial relations between them.

C. Classification

To learn object-action relations from RGB-D videos, we employed a graph network classifier [12] together with the scene graphs returned from our feature extraction pipeline. Graph networks are machine learning building blocks operating on attributes which can be arranged as a graph. Battaglia *et al.* [12] define a graph G as a 3-tuple $G = (u, V, E)$, where u is the global attribute of the graph, V the set of nodes in the graph, and E the set of edges. Each $v_a \in V$ is a node attribute and each $e \in E$ is a 3-tuple $e = (e_a, s, r)$. In this, e_a is the edge attribute, and s and r are the indices of the sender and receiver node in V. A graph network takes such a graph as input, processes it by updating its attributes, and returns it afterwards. The processing takes place in 3 steps, in which following functions are applied: (1) An edge update function ϕ^e; (2) an edge aggregation function $\rho^{e\to v}$ and a node update function ϕ^v; (3) one aggregation function for the nodes $\rho^{v\to u}$ and one for the global attribute $\rho^{e\to u}$, as well as a global update function ϕ^u. This describes a full graph network block, but different types of blocks are possible depending on which update or aggregation functions are used. For example, in

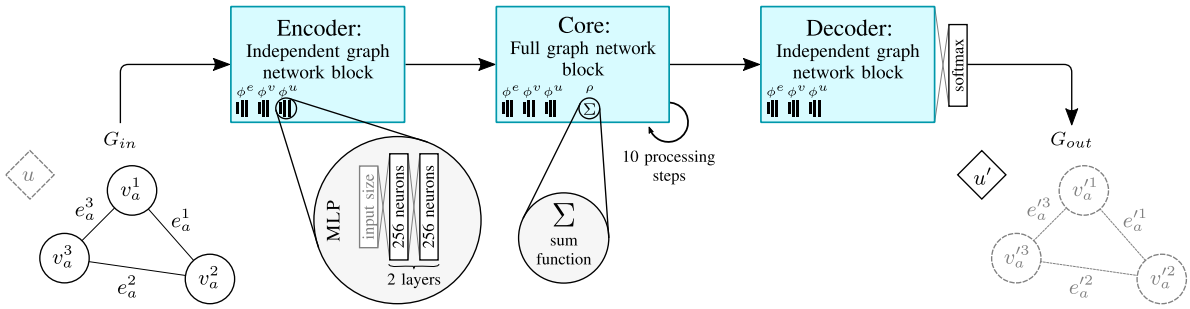

Fig. 4. Architecture of our action classifier, an encode-process-decode graph network with 10 processing steps. The input is a scene graph G_{in} (here exemplarily with 3 nodes) with edge attributes e_a (relations), and node attributes v_a (objects). The global attribute u' of the output graph G_{out} encodes the predicted probability distribution of actions. Grayed out attributes are not used by our classifier. Both the encoder and the decoder, as well as the core, use 3 instances of a multilayer perceptron (MLP) parameterized as depicted for each of their update functions ϕ^e, ϕ^v, and ϕ^u. The core uses the sum function for each aggregation function $\rho \in \{\rho^{e\to v}, \rho^{v\to u}, \rho^{e\to u}\}$. An additional layer is applied right after decoding to scale the latent size to the actual size of u' on the one hand, and apply a softmax to get a probability distribution on the other.

an independent graph network block, no aggregation functions are used. Graph networks can also be composed from several graph network blocks, they do not necessarily have to be atomic graph network blocks. Again, different configurations are possible here as well. For more details about graph networks, we refer to the original publication [12].

We used the *encode-process-decode* configuration for our model, depicted in Fig. 4, for which a reference implementation is available in the Graph Nets library [12]. This model consists of two independent graph network blocks for the encoder and decoder respectively, and a full graph network block for the core. For all 3 blocks, multilayer perceptrons (MLPs) were employed as edge update functions ϕ^e, node update functions ϕ^v, and global update functions ϕ^u. For the aggregation functions $\rho^{e\to v}$, $\rho^{v\to u}$, and $\rho^{e\to u}$, the sum function was used. All MLPs in each graph network block were parameterized with 2 layers and 256 neurons per layer. The core in the encode-process-decode model performed 10 processing steps. These parameters were empirically determined after evaluating multiple test series, each sampling a different configuration. The input of our classifier is a scene graph G_{in}, where the edge attributes e_a encode the relations and the node attributes v_a the object classes. The output is a probability distribution of all actions, which is encoded in the updated global attribute u'. The global attribute u and the updated edge and node attributes e'_a and v'_a are not used.

In our case, all data is symbolic, so one-hot encodings were used for the actions, objects, and relations. The global attribute u encodes the performed action of one hand. This leads to the one-hot encoding $u \in \{0,1\}^{|A|=14}$ for the 14 considered actions. The node attributes $v_a \in V$ encode 12 object classes known to YOLO and one object class per hand. This leads to 14 object classes in total and $v_a \in \{0,1\}^{|O|+2=14}$. All relations are encoded as edge attributes $e_a \in \{0,1\}^{|R|+1=16}$, 15 for the spatial relations, plus one to encode a temporal relation.

Due to noisy depth images and occasional misclassifications from YOLO, certain scene graphs might be ambiguous or not representative for the currently performed action in the frame. To mitigate this effect, we fed a *temporal concatenation* of 10 consecutive scene graphs instead of only one scene graph for the current frame (the current frame plus the 9 previous ones; roughly 333 ms at 30 fps). By temporal concatenation of scene graphs we understand an algorithm, which takes a list of scene graphs as input, and outputs one single scene graph, where all nodes and edges from the input scene graphs are included. The resulting scene graph's global attribute is adopted from the current scene graph while training, but most importantly, *temporal edges* are supplemented by the algorithm. These are edges, which connect the nodes of one specific object instance over a series of frames. Fig. 5 shows an illustration of this process. In other words, a temporal concatenation preserves the number of nodes and edges encoding spatial relations. All nodes connected by spatial relations always belong to one frame, while a path along temporal edges tracks one object instance over multiple frames. Therefore, edges for spatial relations and temporal edges are mutually exclusive. This approach is comparable to how Koppula *et al.* [2], [20] encode temporal relations, however they use it to connect nodes over temporal segments rather than over frames.

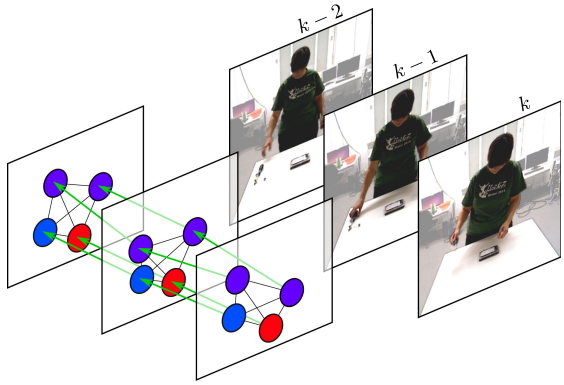

Fig. 5. Example of how a temporal concatenation of scene graphs (left) is constructed (corresponding video frames of each scene graph to the right). For the sake of simplicity, only 3 considered frames ($k-2$, $k-1$, and k) are shown. The temporal edges are depicted in green and trace one object instance over a series of frames.

Conceptually, the classifier is trained on solely the right hand. To account for the left hand, we trained the classifier on *mirrored scene graphs* as well. A mirrored scene graph is a scene graph, where the objects *right hand* and *left hand*, as well as the relations *right of* and *left of* are swapped. Additionally, while training, the ground truth in the global attribute of the target graph is changed to the action performed by the left hand. The benefits of this approach are the reduced training and setup effort, as well as twice as much data through mirroring. At runtime, the scene

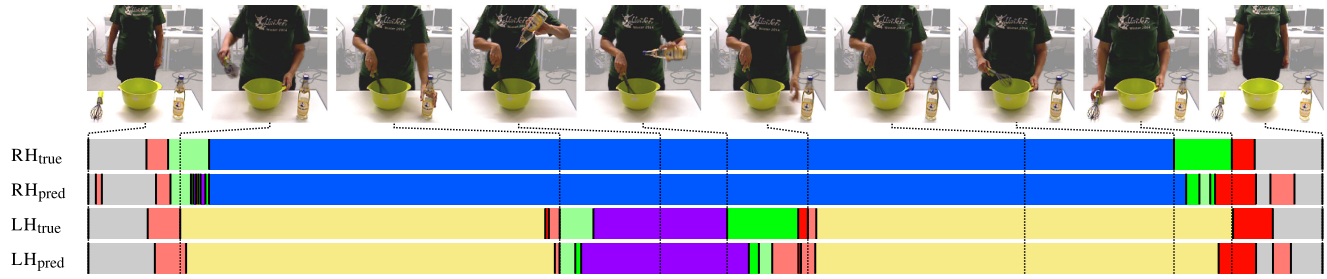

Fig. 6. Qualitative evaluation by visualizing the top prediction of the classifier for the right hand (RH_{pred}) and left hand (LH_{pred}) in each frame over one whole example recording next to the corresponding ground truth (RH_{true} and LH_{true}). Consecutive predictions of the same class were pooled into an action segment of one color. Each color depicts a certain action: idle, approach, retreat, lift, place, stir, hold, and pour.

graph is fed into the network to obtain the action for the right hand. Afterwards, the scene graph will be mirrored and fed into the network again, to get the action for the left.

IV. Evaluation

For the following evaluations, we trained the classifier on our new dataset. For each involved training process, the dataset was split into a training set and a testing set. Testing sets always contained all recordings from one subject, while training sets contained all remaining ones. Additionally, before training, 1 out of the 10 repetitions for each task in the training set were put aside as validation set. We chose a batch size of at most 512 samples, but because of class imbalances, 2 out of 3 samples for the actions *idle* and *hold* were discarded per batch. The Adam optimizer was used with a learning rate of 0.001, and the loss was defined as the cross entropy of the softmax of the global attribute (right hand action) and the ground truth. We stopped the training after the graph network started overfitting and used that state for the evaluation on the test set. The prediction of an action took about 75 ms on an Intel i7 CPU, but can further be improved by utilizing a GPU.

Fig. 6 shows a qualitative evaluation on a recording from subject 1, where the classifier was trained with recordings of the remaining 5 subjects. The top predictions for both hands in each frame were determined, and adjacent predictions were pooled into one action segment. Apart from a few oscillations while lifting the whisk, the classifier was able to form larger contiguous action segments close to the ground truth.

For the quantitative evaluation of the classifier, a leave-one-subject-out cross-validation was performed to obtain 6 folds of training and testing sets. The results of this evaluation are listed in Table I, once for only the top prediction of the classifier, and once again where a prediction was counted as correct if the ground truth was in the top 3 predictions. The latter allows to evaluate, how good the classifier was at identifying correct action candidates. Fig. 7 depicts the normalized confusion matrices, again for the top prediction only, and for the top 3 predictions. As can be seen from the action classification macro F_1 score of 0.86, the classifier is generally able to reliably identify correct candidates. The confusion matrix for the top prediction, however, indicates that in certain cases, it lacks important information to discriminate actions.

A major confusion of the classifier was the prediction of *place* while the true action was *saw*, *pour*, *hammer*, or *drink*. The cause for the prediction of *drink* is that there is no reliable point of reference the classifier could have made use of to distinguish handling a cup (*lifting*, *holding*, or *placing*) from actually *drinking*. Currently, we only consider the human hands, but adding the head to the scene graph could greatly improve the classifier's performance, as *contact* relations would be enough to reliably detect it. The confusions with *saw*, *pour*, or *hammer* can be attributed to wrongly detected 3D bounding boxes. For our feature extraction pipeline, it was especially hard to correctly determine the 3D bounding box for very thin objects like hammers or saws. This can be contributed to our method of estimating them, namely through clustering the part of the depth image outlined by the 2D bounding box and using the biggest cluster. This assumption often fails for thin objects, as the background cluster (mostly the abdomen of the subject) yields more points. Another problem with similar effects was the fact that the bottles (especially the green one) consistently did not yield useful depth information due to high absorption. The described effect can even be observed in the confusion matrix for the top 3 predictions. To mitigate this, it could prove beneficial to replace the bounding box object detection approach with one that yields bounding polygons, which would completely eradicate

TABLE I
Quantitative Evaluation Results. Precision (Precis.), Recall, and F_1 Score of the Action Classification Once for Each Action Class, as Well as Micro, Macro, and Weighted (Weigh.) Averages (avg.) Thereof. Apart From Considering the Top Prediction Only, We Also Evaluated Those Scores Again Where a Classification Result was Considered Correct if the Ground Truth Was in the Top 3 Predictions

	Top prediction			Top 3 predictions		
Action class	Precis.	Recall	F_1	Precis.	Recall	F_1
idle	0.85	0.71	0.78	0.97	0.95	0.96
approach	0.31	0.41	0.35	0.84	0.87	0.86
retreat	0.34	0.43	0.38	0.78	0.84	0.81
lift	0.32	0.50	0.39	0.67	0.82	0.74
place	0.34	0.45	0.38	0.73	0.82	0.77
hold	0.82	0.64	0.72	0.93	0.87	0.90
pour	0.66	0.65	0.66	0.91	0.91	0.91
cut	0.74	0.67	0.70	0.89	0.79	0.83
hammer	0.64	0.56	0.60	0.83	0.75	0.79
saw	0.68	0.58	0.63	0.88	0.72	0.79
stir	0.92	0.84	0.88	0.98	0.97	0.98
screw	0.76	0.79	0.77	0.88	0.89	0.89
drink	0.70	0.71	0.70	0.94	0.94	0.94
wipe	0.78	0.87	0.82	0.92	0.94	0.93
Micro avg.	0.64	0.64	0.64	0.89	0.89	0.89
Macro avg.	0.63	0.63	**0.63**	0.87	0.86	**0.86**
Weigh. avg.	0.69	0.64	0.66	0.89	0.89	0.89

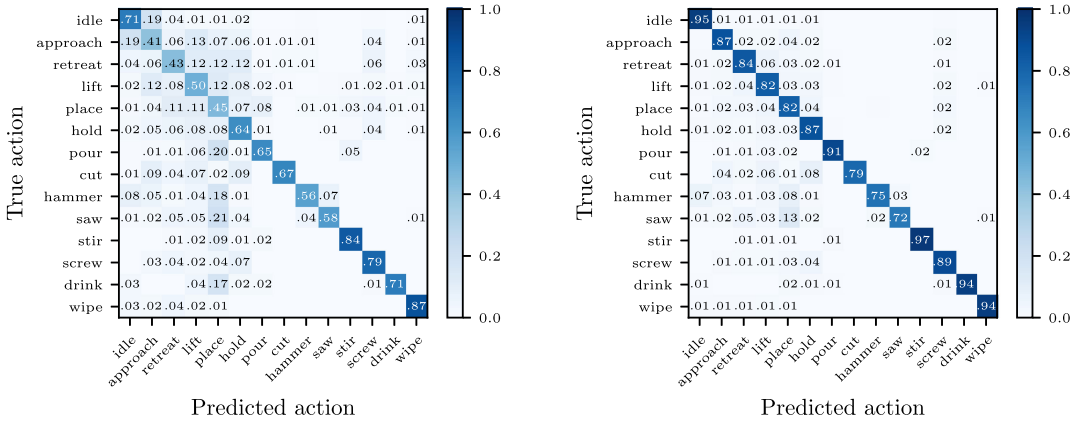

Fig. 7. Accumulative classification correctness over all folds depicted as normalized confusion matrix for the top prediction (left), and where a classification result was considered correct if the ground truth was in the top 3 predictions (right). Empty cells mean no confusion (= 0.00).

the need to perform a clustering in the first place. Additionally, often it would have been helpful to consider the table as object as well to improve the distinction of actions like *lift* and *place*, since the dynamic spatial relations *moving apart* or *getting close* with the table as object would be a strong indicator for either one or the other.

There is also a number of confusions noticeable between the contact-less actions *approach* or *retreat* and the contacting actions *lift*, *place*, or *hold*. The main cause for this is the coarse method to detect contacts, namely collision checks between 3D axis aligned bounding boxes. Additionally, phases of approaching or retreating are usually executed very fast (sometimes within tenths of a second). However, such information is not directly gathered by our feature extraction at this point. Both kinds of confusions could be mitigated by using oriented bounding boxes to estimate the extents of objects.

We also performed the evaluation on the top 3 predictions to show that there is still a lot potential to improve and to motivate future work, as the classifier is generally already able to identify correct candidates for the action. This can be seen in the confusion matrix in Fig. 7 (right) and the action classification macro F_1 score of 0.86. As already pointed out, the top confusions here are the ones involving *place* in conjunction with thin objects, e.g., the prediction of *place* when the true action was *saw* or *hammer*. The comparison to the confusion matrix in Fig. 7 (left) clearly suggests, that the classifier would be able to make better predictions if it was provided with more information of better quality to make the final decision. The suggested improvements to the feature extraction aim at exactly that and could supplement the scene graph in order to provide the classifier with needed information to resolve confusions.

To verify the effectiveness of the architecture and the classifier, an ablation study was performed by removing a set of features, training, and evaluating a new model. First, it was assessed how the classifier performs when only contact relations are considered and all other symbolic spatial relations are discarded. In this case, the action classification macro F_1 score declined to 0.46 compared to the score of 0.63 from Table I, where all spatial relations were considered. This shows that other symbolic spatial relations indeed encode important information the classifier can make use of. Next, the classifier was assessed without considering any spatial relations, and instead encoding the object bounding box centroids in the graph nodes. This

TABLE II

ABLATION STUDY EVALUATION RESULTS. ACTION CLASSIFICATION MACRO F_1 SCORES FOR THE EVALUATIONS PERFORMED IN THE ABLATION STUDY, NAMELY CONSIDERING CONTACT RELATIONS ONLY (CONTACT), CONSIDERING OBJECT CENTROID COORDINATES INSTEAD OF SYMBOLIC RELATIONS (CENTROIDS), AND CONSIDERING NO TEMPORAL RELATIONS (NO TEMP.). THE RESULTS ARE COMPARED TO THOSE ACHIEVED IN OUR QUANTITATIVE EVALUATION (REFERENCE), BOTH FOR THE TOP PREDICTION ONLY (TOP PRED.) AND WHERE A CLASSIFICATION RESULT WAS CONSIDERED CORRECT IF THE GROUND TRUTH WAS IN THE TOP 3 PREDICTIONS (TOP 3 PRED)

	Contact	Centroids	No temp.	Reference
Top pred.	0.46	0.31	0.60	**0.63**
Top 3 pred.	0.73	0.55	0.84	**0.86**

resulted in an action classification macro F_1 score of 0.31, showing that the classifier is not able to derive features of similar quality comparable to the symbolic relations, and that *contact* relations alone are valuable information. Finally, an evaluation on raw scene graphs without any temporal concatenations was performed, to assess, how beneficial they are. This resulted in an action classification macro F_1 score of 0.60. Even though the classifier performed better with temporal relations, the improvement was rather minor considering the up to 10 times higher processing effort for training and execution. The results of this ablation study are listed in Table II.

A direct comparison to approaches from the literature proved challenging, because, to the best of our knowledge, there are no bimanual action recognition approaches to compare with. Considering only one hand is not meaningful, because this is not the same as the action labels for the overall prevalent action as seen in most other approaches and datasets. Additionally, YOLO currently limits us to our dataset, as we trained it to specifically recognize the objects occurring in that only.

V. CONCLUSION AND FUTURE WORK

In this work, we presented an approach to learn object-action relations from bimanual human demonstration, for action segmentation and recognition. The proposed classifier takes scene graphs as input, and provides bimanual predictions of the performed actions. Using a graph network allows us to encode a scene without having to consider the amount or order of the objects. Additionally, we do not require any prior action

segmentation at this point. To obtain the scene graphs from RGB-D video frames, we developed a feature extraction pipeline making use of two state-of-the-art vision frameworks to detect objects and the human teacher's pose to obtain 3D symbolic spatial relations between objects and hands. Other than that, we contribute a novel RGB-D dataset of subjects performing bimanual actions in the kitchen and workshop. The ground truth action labels are provided on a per-hand basis.

In future work, including the table as global point of reference and the head of the teacher could further improve the prediction quality. The pose information of the human teacher is available and could be used to account for the whole upper body movement. The biggest negative impact on the prediction quality, however, can be attributed to wrong relations resulting from misplaced 3D bounding boxes. At this point, we also only consider 10 frames (≈333 ms) to predict the action in the most recent frame, but long term sequence information could be important data for the classifier, as certain actions follow a logical sequence (e.g., *lift* implies *place* later on).

REFERENCES

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot Programming by Demonstration," in *Handbook of Robotics*. Berlin, Germany: Springer, 2008, pp. 1371–1394.

[2] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 14–29, Jan. 2016.

[3] F. Ziaeetabar, T. Kulvicius, M. Tamosiunaite, and F. Wörgötter, "Recognition and prediction of manipulation actions using enriched semantic event chains," *Robot. Auton. Syst.*, vol. 110, pp. 173–188, Dec. 2018.

[4] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[5] H. Kjellström, J. Romero, and D. Kragić, "Visual object-action recognition: Inferring object affordances from human demonstration," *Comput. Vision Image Understanding*, vol. 115, no. 1, pp. 81–90, Jan. 2011.

[6] M. D. Rodriguez, J. Ahmed, and M. Shah, "Action MACH: A spatio-temporal maximum average correlation height filter for action recognition," in *Proc. Conf. Comput. Vision Pattern Recognit.*, Jun. 2008, pp. 1–8.

[7] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Proc. Int. Conf. Comput. Vision*, vol. 2., Oct. 2005, pp. 1395–1402.

[8] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object-action relations by observation," *Int. J. Robot. Res.*, vol. 30, no. 10, pp. 1229–1249, Sep. 2011.

[9] M. Wächter and T. Asfour, "Hierarchical segmentation of manipulation actions based on object relations and motion characteristics," in *Proc. Int. Conf. Adv. Robot.*, Jul. 2015, pp. 549–556.

[10] C. Mandery, J. Borrás, M. Jöchner, and T. Asfour, "Analyzing whole-body pose transitions in multi-contact motions," in *Proc. Int. Conf. Humanoid Robots (Humanoids)*, Nov. 2015, pp. 1020–1027.

[11] R. Zöllner, T. Asfour, and R. Dillmann, "Programming by demonstration: Dual-arm manipulation tasks for humanoid robots," in *Proc. Int. Conf. Intell. Robots Syst.*, vol. 1, Sep. 2004, pp. 479–484.

[12] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," Jun. 2018, *arXiv: 1806.01261* [cs, stat].

[13] H. Kühne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *Proc. Conf. Comput. Vision Pattern Recognit.*, Jun. 2014, pp. 780–787.

[14] D. Damen *et al.*, "Scaling egocentric vision: The EPIC-KITCHENS dataset," in *Proc. Eur. Conf. Comput. Vision* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2018, pp. 753–771.

[15] L. Zhou, C. Xu, and J. J. Corso, "Towards automatic learning of procedures from web instructional videos," in *Proc. AAAI Conf. Artif. Intell.* AAAI Press, Apr. 2018, pp. 7590–7598.

[16] C. Wu, J. Zhang, S. Savarese, and A. Saxena, "Watch-n-Patch: Unsupervised understanding of actions and relations," in *Proc. Conf. Comput. Vision Pattern Recognit.*, Jun. 2015, pp. 4362–4370.

[17] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Human activity detection from RGBD images," in *Proc. AAAI Conf. Artif. Intell. Workshops*, Aug. 2011.

[18] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Proc. Conf. Comput. Vision Pattern Recognit.*, Jun. 2012, pp. 1290–1297.

[19] L. Xia, C.-C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3D joints," in *Proc. Conf. Comput. Vision Pattern Recognit. Workshops*, Jun. 2012, pp. 20–27.

[20] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from RGB-D videos," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 951–970, Jul. 2013.

[21] E. E. Aksoy, M. Tamosiunaite, and F. Wörgötter, "Model-free incremental learning of the semantics of manipulation actions," *Robot. Auton. Syst.*, vol. 71, pp. 118–133, Sep. 2015.

[22] M. Tenorth, J. Bandouch, and M. Beetz, "The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition," in *Proc. Int. Conf. Comput. Vision Workshops*, Sep. 2009, pp. 1089–1096.

[23] R. Poppe, "A survey on vision-based human action recognition," *Image Vision Comput.*, vol. 28, no. 6, pp. 976–990, Jun. 2010.

[24] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Comput. Vision Image Understanding*, vol. 115, no. 2, pp. 224–241, Feb. 2011.

[25] J. M. Chaquet, E. J. Carmona, and A. Fernández-Caballero, "A survey of video datasets for human action and activity recognition," *Comput. Vision Image Understanding*, vol. 117, no. 6, pp. 633–659, Jun. 2013.

[26] J. Zhang, W. Li, P. O. Ogunbona, P. Wang, and C. Tang, "RGB-D-based action recognition datasets: A survey," *Pattern Recognit.*, vol. 60, pp. 86–105, Dec. 2016.

[27] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image Vision Comput.*, vol. 60, pp. 4–21, Apr. 2017.

[28] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour, "The robot software framework ArmarX," *IT–Inform. Technol.*, vol. 57, no. 2, pp. 99–111, Mar. 2015.

[29] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," Apr. 2018, *arXiv: 1804.02767* [cs].

[30] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," Dec. 2018, *arXiv: 1812.08008* [cs].