Detecting Errors in Pick and Place Procedures

Detecting Errors in Multi-Stage and Sequence-Constrained Manual Retrieve-Assembly Procedures

Riccardo Bovo†
Department of Computing
Imperial College London
London, UK
r.bovo19@imperial.ac.uk

Nicola Binetti
UCL Interaction Center
University College London
London UK
n.binetti@ucl.ac.uk

Duncan P. Brumby
UCL Interaction Center
University College London
London UK
d.brumby@ucl.ac.uk

Simon Julier
Department of Computer Science
University College London
London, UK
s.julier@ucl.ac.uk

ABSTRACT

Many human activities, such as manufacturing and assembly, are sequence-constrained procedural tasks (SPTs): they consist of a series of steps that must be executed in a specific spatial/temporal order. However, these tasks can be error prone - steps can be missed out, executed out-of-order, and repeated. The ability to automatically predict if a person is about to commit an error could greatly help in these cases. The prediction could be used, for example, to provide feedback to prevent mistakes or mitigate their effects. In this paper, we present a novel approach for real-time error prediction for multi-step sequence tasks which uses a minimum viable set of behavioural signals. We have three main contributions. The first we present an architecture for real-time error prediction based on task tracking and intent prediction. The second is to explore the effectiveness of using hand position and eye-gaze tracking for task tracking. We confirm that eye-gaze is more effective for intent prediction, hand tracking is more accurate for task tracking and that combining the two provides the best overall response. We show that using Hands and Gaze tracking data we can predict selection/placement errors with an F1 score of 97%, approximately 300ms before the error would occur. Finally, we discuss the application of this hand-gaze error detection architecture used in conjunction with head-mounted AR displays, to support industrial manual assembly.

CCS CONCEPTS

• Human-centered computing → User models; Interaction paradigms; Laboratory Experiments •Computing methodologies → Neural Networks

†work carried out while at UCL Interaction Center

IUI '20, March 17–20, 2020, Cagliari, Italy
ACM ISBN 978-1-4503-7118-6/20/03…$15.00
https://doi.org/10.1145/3377325.3377497

KEYWORDS

Manual Assembly Procedures, Error Prediction, Human-centered design, User Intent prediction, Long-Short Term Memory, Intelligent Assistive Systems.

ACM Reference format:

Riccardo Bovo, Nicola Binetti, Duncan P. Brumby and Simon Julier. 2020. Detecting Errors in Pick and Place Procedures: Detecting Errors in Multi-Stage and Sequence Constrained Manual Retrieve-Assembly Procedures. In *Proceedings of IUI conference (IUI'20). ACM, New York, NY, USA, 10 pages.* https://doi.org/10.1145/3377325.3377497

1. Introduction

The assembly of a robot part on a factory floor, or the servicing and maintenance of a machine, are both examples of Sequence-constrained Procedural Tasks (SPTs). These tasks consist of a set of steps that must be carried out in a specific temporal order. However, following an SPT can be error-prone: steps can be carried out in the wrong order, they can be missed out or they can be repeated. These errors normally have negative consequences, such as reducing efficiency, increasing waste of resources and reinforcing negative behaviours such as a sense of frustration. Therefore, effective and efficient approaches for real time error detection could help many maintenance and assembly tasks. These could be used, for example, to provide real-time feedback to inform a person if a task has been carried out correctly. More importantly, if the system can predict that an error might occur, it could prevent that error from occurring in the first place or allow the operator to fix the error as soon as it occurs.

Error prediction in an SPT depends upon three pieces of information: the current step the user is working on (step tracking), the appropriate set of actions which can be taken in that step (predefined sequence of steps), and the action the user is about to undertake (user intent prediction). While some studies [16, 18, 22] cover a subset of the real-time error prediction problem, to the best of our knowledge there is no study that covers all three of them.

Our contribution is threefold. First, we develop a new architecture for intent tracking and error prediction. Our

architecture uses a comprehensive model to describe activities. Rather than use a set of heuristics, we use a machine learned approach to learn the relationships between signals and both the current step and predicted activities. This learned model predicts the coordinates where parts will be picked and placed and does not depend on a specific action sequence. Second, we explored the relative merits of using hand and eye gaze signals to both track the step and infer user intent. We compared performance of the system *using hand signals only, eye signals only, and both hand and gaze* signals in parallel (I.e. both signals used to track procedural steps and infer user intent) or in series (i.e. hand signal used to track procedural steps and eye-gaze to infer intent). These results confirm the different roles played by different signals and highlight an effective configuration for error prediction consisting on the combined use of hand and eye-gaze to predict intent and the use of hand alone to track the procedure.

Third, we discuss applications of the proposed system in conjunctions with head-mounted AR displays. We highlight how the signals of hand and eye-gaze are consistent with signals available thought HMD AR. We outline how such error prediction system could be used as an intelligent assistive system to control (display/forwarding) of assembly instruction.

We created an experiment to record eye and hand behaviour of participants performing repetitions of a pick-and-place building block task. This task has been used in previous research and is considered a valid abstraction of industrial assembly task [2, 6, 15, 20]. Building block number and assembly instructions were manipulated across experimental blocks to introduce different levels of task complexity / uncertainty, with the assumption that greater task difficulty would increase the likelihood of observing procedural errors. We used hand and gaze data, labelled according to onsets of grasping, picking, placing, releasing hand behaviours, to train the LSTM RNNs.

We provide a real-time prediction architecture which uses a minimum viable set of behavioural signals (gaze, hand), to predict selection/placement errors with an accuracy of 97%, approximately 300ms preceding response onset.

2 Problem Statement

2.1 The SPT model

We consider the problem of a user performing an assembly task in a manufacturing context. The correct manufacturing procedure is the SPT T_p. T_p is an ordered set of n steps:

$$T = (S_1, S, \dots, S_n) \quad (1)$$

In each step S_i, a worker obtains apart from a *resource area* and places moves it in to an *assembly area* where an assembly operation is carried out. Therefore, each step consists of a single retrieve-assembly pair:

$$S_i = (R_i, A_i) \quad (2)$$

Both are spatially anchored. For each R_i there is a retrieve location r_i which specifies where the part is obtained from; and for each A_i the assembly location a_i specifies where the part is placed. In many assembly tasks, these locations are fixed – parts are taken from a bin or location and fitted to a well-defined point on the assembly. Although it is convenient to specify the SPT in terms of (R_i, A_i), this is not sufficient to perform error detection because it does not specify the detectable actions a worker will make. Instead, we decompose each step further.

2.2 Step Decomposition

Each step is decomposed into a set of actions which can be measured by the gaze and hand trackers. We use the General Assembly Task Model (GATM) proposed by Funk [6]. GATM is a general model of assembly tasks and can be used to develop standardized experiment for the design and evaluation of assembly instructions. Therefore, it offers a solid base to model an error prediction system for SPTs. The GATM decomposes each assembly step into four phases:

1. *Reaching:* starts when the worker reaches towards an assembly part and ends when the worker touches the part.
2. *Grasping:* starts when the worker touches an assembly part and ends when the assembly part is about to be lifted from the resource area.
3. *Placing:* starts with the lift of the assembly part from the resource area and ends when the assembly part touches the assembled surface.
4. *Releasing:* starts when the assembly part touches the assembly area and ends the moment the worker's hand detaches from the assembly part.

These phases directly map to the retrieve-assembly pairs. *Reaching* and *grasping* form the retrieve action while *locate* and *place* constitute the assembly action. Using this decomposition, we represent the current state of an SPT at time t as the vector $x_t = (s_t, p_t)$, where s_t is the step number (an integer in the range $1, \dots, n$) and p_t is the phase number in the step (an integer $1, \dots, 4$ corresponding to the four phases above).

2.3 Predictive Error Detection in the SPT model

Assembly errors are defined in terms of a mismatch between the specified manufacturing procedure T_p and the procedure the user actually carried out. Given the GATM, it is impossible for a worker to skip a phase (for example, a part cannot be placed in the assembly area until it has been grasped). Therefore, errors arise if the user performs a retrieve or assembly operation which is inconsistent with the spatial location specified by the manufacturing procedure. To do this, the system must be able to:

1. Track the task and estimate x_t.
2. Given x_t, determine what is the next valid set of actions and where they will occur.
3. Infer the actions the user is undertaking and check if the locations are consistent with the current step and phase.

Predictive error detection extends the third capability by predicting, at some future time, the action the user will undertake. We now

review the literature associated with error detection and intent prediction.

3 Related Work

As noted above, detecting an error with an SPT includes both tracking the current state x_t and predicting future actions.

3.1 Task Tracking

Task tracking estimates the current step a user is on. Since SPTs are linear sequences, Finite State Machines (FSMs) can be used. Sensor data is recorded and, when certain conditions are met, the system triggers a transition to a new phase. However, this assumes that the phase transitions are detected perfectly. Sensor noise and imperfections in the detector mean this not always the case. Therefore, Hidden Markov Models (HMMs) are often used [3] to model the probability distribution over the phase. Whether HMMs are used or not, there are two broad approaches: tagging-based approaches, and remote monitoring.

Tagging methods instrument the workers and / or the environment with sensors to monitor progress. For example, in [15] the environment and all of the tools are tagged with RFID tags which detect proximity [23] uses sensors on the worker. The system detects discrete events from three sensors: an RFID reader, a force reed and an inertial sensor. However, explicit tagging techniques have the difficulty that they can become highly intrusive and very cumbersome.

An alternative approach is to use sensors which remotely monitor the assembly process without the need for tagging. For example, in [15] a web camera was used to detect part and tools positions/configuration, and an ultrasound beacon system was used to track hand positions. While this approach avoids the need for tagging, it relies on recognising each individual part, and the part configurations. As a result, for each new assembly task, a new system has to be retrained. Therefore, an approach that relies on recognition assembly procedure's common features (humans body parts and actions) is potentially more generalizable. Furthermore, as noted above, in our manufacturing problems assembly errors are defined by errors in location rather than in action.

One approach is to use body-posture to detect actions in manual procedures [18, 22, 24]. However, in many cases, it is likely that using full body pose used to predict manual activities contains information that is redundant. In the same way, when developing a real-time system it is important to optimize the amount of data used to make detection/prediction in order to reduce the computational effort. We contribute by testing and measuring the effectiveness of the hand signal (single joint) as an alternative to the full body pose (~15 joints) when predicting user intent or detecting actions within a manual task.

3.2 Intent Prediction with Eye Gaze

Most work in the literature has explored the related problem of predicting user intent from behavioral signals For example, eye gaze has been used to anticipate user intent in collaborative tasks between teams of humans [11], and between teams of humans and robots [1, 10, 21], in digital-screen procedures [16], and to assist upper limb amputees with grasp activities [7]. These studies use gaze as a proxy to the user's thinking process. They can be used to identify a user's mental states (hesitation, confusion, clarity) [11] and intention (i.e. which object will be selected) [1, 10, 21]. Physiological studies have shown that eye gaze precedes action [4, 5, 14]. Therefore, several authors have used this fact to predict intended action. [4] ran a study about user intention prediction using just eye gaze tracking. A set of participants performed a screen-based procedure; within it a single action step was observed and measured. Eye tracking data was collected and the number of eye gaze fixations over the specific screen area was counted; a logistic regression was then trained and tested. Results shows a 75% accuracy of the model in predicting if the user would perform a target action.

[3] ran a study which used eye tracking (gaze signal) as the input to a support vector machine (SVM). The SVM predicted user intent in a collaborative assembly tasks between humans. A series of participants performed a sandwich making task in which a "worker" prepared a sandwich by adding the ingredients requested by a "customer"; the behavioural data (verbal and gaze) of the latter is collected. The data was then segmented into selection segments based on verbal behaviour (i.e. customer expressing ingredient preferences) and used to train an SVM model. Results shows that the model achieved 76% accuracy in predicting customer intended request only using gaze features; additionally, such correct prediction was made on average 1.8 s before the spoken request. A limitation of these two studies is that eye-gaze was used to predict user intent in a non-manual task, therefore it might not generalize to visually guided manipulation tasks and assembly procedures. A second limitation of the approach in [3] is that the system had no way to automatically infer end/start of a procedural step. Rather, the user manually notified the system.

Some of these studies [11],[3]–[5], use eye-gaze to predict user intent within multiple-action sequences; however none explore the tracking of such sequences (i.e. detecting start end of an action and recording the sequence). We contribute by testing and measuring the effectiveness of the gaze used in isolation to keep track of a manual assembly task (i.e. detecting start end of actions within a sequence of actions).

3.3 Intent Prediction over Multiple Steps

Ravichander et al. [18] [17] ran a study focused on predicting multiple action in unconstrained sequences (i.e. without a correct/incorrect reference sequence). They collected body-pose data from two participants assembling an amplifier. The amplifier assembly task has multiple correct assembly sequences, and so it is not sequence constrained. Using the body pose data, [18] developed and tested the G-MMIE algorithm. This inferred both the target object a user was reaching for, as well as detecting the completion of the user reaching motion. The end of the reaching motion was used to switch to the next assembly step [17]. A limitation of G-MMIE is that it is based on a definition of 'action' that is limited to just the reaching event, and so it does not account for its environmental consequence (such as grasping versus not grasping

an object and therefore switching to the next procedural steps vs remaining in the same step).

[9] also ran a study focused on intention prediction for multiple actions-steps in unconstrained sequences (i.e. without a correct/incorrect reference sequence) using both skeletal data and eye gaze to predict intent. An intention recognition model was developed (six actions classes) using an LSTM RNN. Predictions were carried out using an LSTM RNN with an Encoder Decoder architecture. The first model (i.e. user intention recognition) was trained and tested within a series of single action sequences using as input the single body pose as well as a combined body and eye tracking pose which results in higher accuracy for the combined input. The prediction model was trained and tested on multiple action sequences. A limitation of this approach is that it does not keep track of the completion of a procedure.

3.4 Knowledge Gap

Within the literature there are a few examples of studies that can be insightful for error detection: such as intention recognition within multistage manual assembly procedures using behavioural signals [17, 18, 22], and procedures tracking using assembly part recognition [15]. But to the best of our knowledge previous work has not attempted to combine procedure tracking with intention recognition in the context of a generalizable retrieve-assembly procedure. Additionally, no studies exist which uses a common set of inputs to achieve both procedure tracking and intention recognition. Ultimately there is no example of studies which models human assembly errors (HAE) and aims to detect HAE within a general assembly task. The challenge is therefore to model an error detection system generalizable to most retrieve/assembly tasks which parts uses a minimal set of inputs. Such real-time error detection is desirable and important as it will allow to trigger a feedback to the user in order to mitigate the detected error.

While within literature there are approaches that uses simple heuristics [21], we choose to use an AI model because it does not require the engineering of specific features for each of the signal used (i.e. gaze patterns might be completely different from to be recognized). We use the LSTM model as a black-box to capture nonlinear variations within and between the behavioural signals in order to quantitatively evaluate the efficiency of each signal. Our approach uses a set of machines learning algorithms. First, using an LSTM-based approach we detect switches between the stages of a GATM, and keep track of the assembly procedure by returning the current procedural step of an STP. Second, we implement an LSTM-based user intent predictor that infers the location of a target part and target assembly-location for that part. Third we develop a comparison module between the current intent and the correct action for the current step.

4 Implementation of the Error Prediction System

The architecture for our error detection system is shown in Figure 1. It consists of four main components: measurements, task tracking, intention detection and error prediction. We describe each component in turn.

4.1.1 Measurements: The measurement system is shown in Figure 2. It consists of the input gaze and hand signal measurements. The gaze consists of the (x, y) coordinates of the intersection of the gaze ray on a workspace. Similar, the hand consists of the (x, y) coordinates of the centroid of the hand projected onto the workspace. The history of the measurement sequence is stored and inference/prediction are carried out over a sliding window of the last 2 seconds of measurements. The 2 seconds time length has been set empirically, and it provide the network with enough data for it so that movements features can be recognized. The 2s length it is not a time constrain for the movement execution (i.e. movements duration does not influence their detectability).

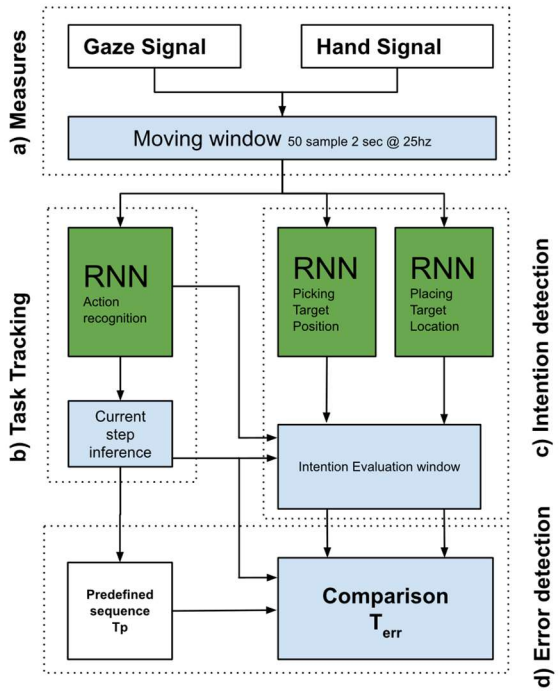

Figure 1 - System Architecture on the right containing a) Measures b) Task Tracking and c) Intention detection d) Error Detection

4.1.2 Task tracking: The task tracking component estimates the state $x_t = (s_t, p_t)$. It is implemented using the phase detection LSTM RNN. This recognizes the low-level phases (rc, gr, pl, rl) from the measurement window. When the release phase of one step is completed, the task tracker moves to the next step in the SPT and the reaching phase is started. The phase detector occasionally makes mistakes including detecting a phase which does not occur, and missing phases which happen. In the case of a missing step the task tracker moves to the next step even in the event of sampling an incomplete step sequence: for example (pl, rl, rc) triggers a move

even if the grasp event is missing, another example could be (gr, pl, rc) or (pl, rl, gr) missing respectively the release and reach event. In the case of the low-level phase inference of an event that does not occur the strategy we adopt is to delete such event: for example, (rc, gr, pl, rc, rl). This logic holds as our dataset only contains complete movements as we constrain it at the experiment level.

4.1.3 Intention detection: The LSTM RNN predicts the target location of a retrieve/assembly action. Although we initially tried to implement a single predictor which could be applied to both actions, we found that developing predictors for each phase separately gave better performance. Both predictors use the windows of gaze and hand information to predict the spatial location where the hand will come to rest when the user is retrieving or placing piece. This component starts predicting when the user first enters the reaching/placing phase and continues until the end of the phase. The predictions can be very noisy over time. Therefore, the evaluation window component evaluates the sequence of outputs of the picking/placing target position predictor over the duration of the phase (reaching/placing phase duration). The evaluation is triggered at the end of the reaching/placing phase and consist in counting the max predicted location within the phase window.

4.1.4 Error prediction: Because the steps and phases are constrained, errors arise only if the locations are incorrect. A retrieve error happens when the selected part has a different location from the correspondent step instruction; while an assembly error happens if the block is left on a different position from the one specified by the correspondent step instruction. Such comparison is triggered once the intention recognition confidence exceeds a threshold or, in case the threshold is not exceeded, is triggered at assembly time (transition between rc, gr) or retrieve time (transition between pl, rl), see Fig. 1. If intention confidence is not reached than comparison between predefined/detected steps happens at the start of either grasp/release phases. Since such comparison happens before the end of retrieve/assembly action (i.e. end of grasp/release phase), we can say that if an error happens is always detected before the completion of the action (i.e. predicted).

5 Dataset

In this section we explain how we generated the dataset used to train and test the system and how we manipulated complexity and instructions throughout the experiment to generate behavioural variability and errors within the dataset.

5.1 Data Collection

In order to study people's performance on manufacturing assembly tasks, a number of previous studies have used a lab-based Duplo pick-and-place task [2, 6, 15, 20]. This task captures the essential cognitive and behavioural components of procedural assembly work while requiring a shorter assembly time than an industrial retrieve-assembly task [6].

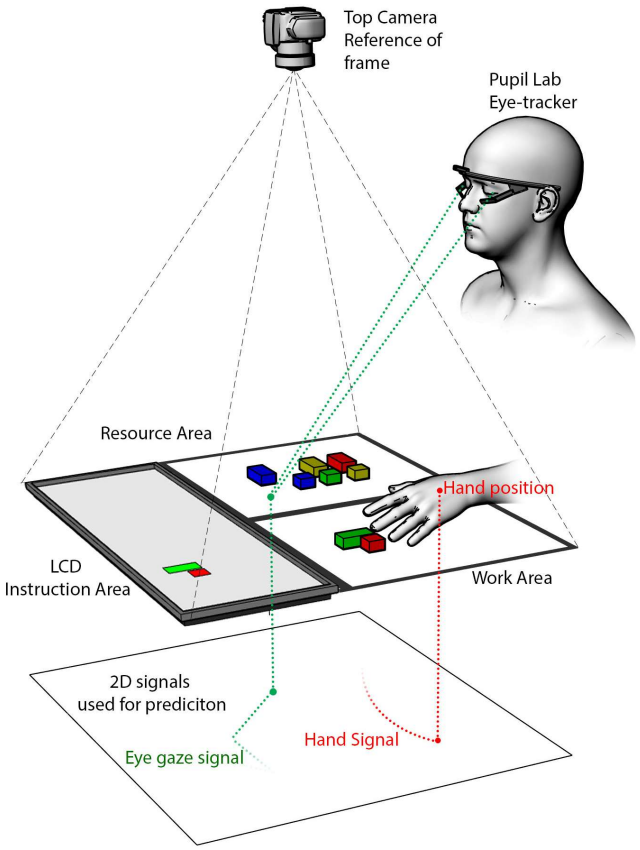

Figure 2 – a) Experiment Set-up. The layout has three areas (resource, workspace, area). A top camera is used to monitor activity of the user's hands and the blocks. The wearable eye-tracker measures the user's gaze.

5.2.1 Participants: 12 participants (7 males) were recruited from a university participation pool. Some participants were students, and some were teaching staff, and their ages ranged between 24-47 (average 32.6 +/- 6.8 years). Each participant was compensated with a £7.50 Amazon voucher. All participant had normal or corrected-to-normal vision.

5.2.2 Task and Layout: the experiment layout is modelled after [14]. It is shown in Figure 2 and consists of three areas: a *resource area* (from which blocks were retrieved), a *workspace area* (in which blocks were placed), and the *model area* (where the sequence instructions were displayed). The resource area is additionally divided into specific *Bins* (Fig 3) and the *work area* is also additionally subdivided by a *coordinates grid* (Fig 3).

The experiment task consists on a Duplo block pick-and-place tasks by following instructions (step-by-step pictorial depiction of steps). Instructions are shown for each step on *the LCD Instructions Area*, blocks are initially placed into specific *Bins* in the *Resource Area*. Participants follows instructions and moves a specific block (unique combination of colour and size) from the *Resource Area* to a specific coordinate in the *Work Area* (Fig 2 and Fig 3). Instructions are shown alternated across even and odds trials

so to induce errors (see paragraphs 5.2). Easy Task consists in a 4-block sequence and the complex task in a 8-block sequence.

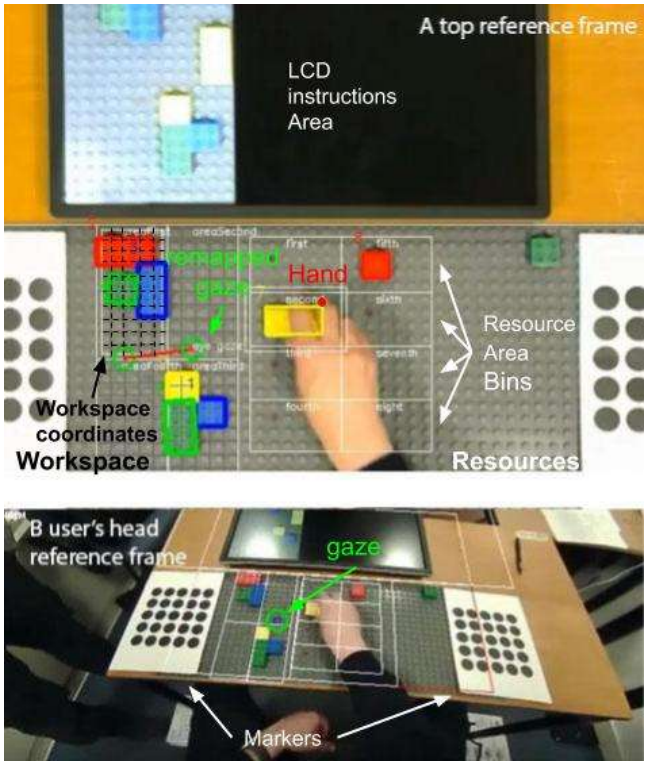

Figure 3 – Video feed of the A) "top reference frame" and below B) "head reference frame" with the registered eye-gaze and hand signals. A) "8 Resource Bins" can be seen, each one of them contains a building block at the beginning of every task, "workspace coordinates" can be seen on the left side of the picture which (black dotted lines), "remapped gaze" can be seen in green "hand signal" can be seen in red. B) the below image shows the user point of view taken from the head mounted pupil lab eye-tracker, as well as the detected gaze signal in green. On both A) and B) is possible to see the markers (black dots on white background) these are placed on the left and right side of the resource and assembly areas. Markers are used to remap the gaze signal from the user reference frame to the top reference camera where the block detection system and hand detection system are.

5.2.3 Procedure: The experiment procedure consists in the participant routinely perform the Duplo block pick-and-place tasks by following instructions (step-by-step pictorial depiction of steps). The instructions were delivered on the first three trials, and successively alternated across even and odd trials (no instructions on the 4th trial, instructions available on the 5th, etc). Emphasis was placed on remembering the procedure in the correct order and executing it as quickly as possible, but without sacrificing accuracy. After completing each procedure, we reset the blocks by repositioning them randomly on the resource area (so that the blocks' initial positions changed across the trials). Resetting of the blocks was done as quickly as possible to minimise any potential memorising activity that participants could carry out by observing the final block configuration. Subsequently, subjects were asked to perform the next experimental block. The entire experiment took approximately 40 min. The total Number of usable trials collect was 118 with a comprehensive number of steps of 797.

5.2.4 Materials: During the experiment the gaze (eye tracking data) and videos of the working area were recorded. A building block detection system and a hand detection system were implemented using the Open Computer Vision (OpenCV) library. Using the video feed from the top view stationary webcam shown in Figure 3, the block detection systems reconstructed participant's assembling behaviours labelling the dataset with the correct step number, while the hand detection system recorded the hand motion. The camera has a resolution of 640x480px, a latency of 65ms and the block detection algorithm has a precision of 5 pixels rms. Eye tracking data was collected by a head mounted Pupil Labs eye tracking system (https://pupil-labs.com/) recording eye-gaze position at 90Hz with an angular accuracy 0.6 degree and a temporal latency of 45ms. Markers were used to spatially register the coordinate frames of the over-head camera and the eye gaze cameras. The collected dataset was resampled with a linear interpolation at a frequency of 25Hz. Behaviour was recorded as four-dimensional trajectories over the experiment area (hand/gaze signal timeseries). Gaze and hand behaviour were represented as times-series of fixation and hand barycentre positions in 2D space.

5.2 Inducing retrieve/assembly errors

We used two independent variables to vary the difficulty of the task: *presence of instructions* and *level of complexity*. We manipulated *presence/absence of instructions* to generate two fundamentally different modality of procedure execution. In the presence of instructions, a participant performed the procedure using an external model/reference; whereas without instructions a participant had to rely on their internal mental model of the procedure.

During the experiment procedure we interweaved (across procedure repetitions) *presence/absence of instructions*. This made it possible for the participant to build (across repetitions) an internal mental model of the procedure. When a participant used his/her internal mental model while this was not yet complete/solid he/she committed procedural errors.

Each experimental section consisted of 10 trials. In each trial the participant had to fully assemble the block configuration. Easy/hard experimental sections were counterbalanced across participants to remove skill learning artefacts (e.g., learning layout anchor points).

Complexity and *instruction* levels were manipulated to induce differences and variability of participant performance measures and hand/gaze signal timeseries. Greater task complexity and absence of instructions introduce uncertainty in task performance, task sequence internalization and greater chances of procedural errors.

For the 4 different combinations of *Complexity* and *instruction* we outline the average percentage of errors measured: easy task with instructions 5%, easy task without instructions 25% errors,

difficult task with instructions 1.25%, difficult task without instructions 50%. The total number of movements recorded is 1200 movements of which 26.5% -318 errors (incorrect movements) and 73.5% - 882 correct.

5.3 Implementation and Training

The architecture in Figure 1 uses three RNNs: *Low-Level Action Inference, Picking Target Position* and *Placing Target Position.* Each of these maps a sequence of input features to a single categorical output: p_t, r_t, a_t for every value of t; where: p_t consists in the detected phase at time t, r_t consists in the predicted target retrieval area at time t. and a_t consists in the predicted assembly target area at time t. Each network is implemented using a Many-to-One architecture and all take a inputs which consist of *Hand*, *Gaze* or *Hand+Gaze*. Both the *Picking Target Position* and *Placing Target Position* taken an additional 16 dimensions (8 x, y tuples, where 8 is the max number of blocks) defining either the initial position of the blocks (for picking) or the target location of the blocks (for placing). Such information was explicitly passed to the RNN as block initial location was randomized at every trial and because the assembly target location changed across the easy-complex task (experiment manipulation section 5.2).

The *Low-Level Action Inference* LSTM RNN was implemented using Keras and TensorFlow. The chosen optimizer was RMSProp (Root Mean Square Propagation) and the loss function was categorical cross entropy. To validate the model, we used a leave-one-participants-out cross validation: the training set consisted of the data of 9 participants (800 block movements of which 212 errors and 588 correct) while the validation set consisted of the remaining 3 participants data (400 blocks movement of which 80 errors and 320 correct). The training dataset contained 45,846 samples (moving windows) and was validated using 13,941 samples. *Picking and Placing target position* were trained using a dataset of 16,695 samples and validated using 2,800 samples. The layers consisted of:

1. **Input Layer:** The input layer dimension varies accordingly to the measure used (H*and*, *Gaze* or *Hand+Gaze*) In general the input looked like (n of timestep, sequence-size, features dimensions). Where the timestep number represent the duration of the task, the sequence size consists in the moving window dimension (50 sample equal to 2 sec@25hz) and the features dimension depend on the combination of hand or gaze signal used. When either gaze or gesture data was used, the dimension is 2. When both are used together, the dimension is 4.
2. **LSTM Layer:** has a memory component that carried information across the window timesteps. The many-to-one architecture consist in the output to loses the time dimension (return sequence = false). The output space dimension of this layer is 128.
3. **Dropout Layer**: dropped 20% of the LSTM output to prevent the model from overfitting.
4. **Dense Layer**: a fully connected layer with a linear activation and an output space dimension of 64.
5. **Dense Layer:** a fully connected layer with a linear activation and an output space dimension of 12.
6. **Output Dense Layer**: a fully connected layer with a SoftMax activation and an output dimension equal to the number of classes. This is either 4 (low-level phases recognition) or 8 (intention recognition).

We measured the computational cost of the RNNs to understand its impacts on prediction time. We did so by measuring its execution time running the 3 RNN's with a NVIDIA TITAN X GPU. The average inference time recorded was 11ms (8ms, 14ms, 11ms).

6 Results

6.1 Choice of Inputs for Each Network

In order to define a minimal set of input signals for the system and its subcomponents we trained the LSTMs for both *Task tracking* and *Intent detection* with *Hand* only*, Gaze* only or *Hand+Gaze*. To test the performance, we used the measurement:

$$accuracy = \frac{correctly\_identified\_samples}{total\_samples}$$

Since the purpose of *Task tracking* is to support the detection of retrieve and assembly errors, we defined *total samples* to be the total number of times the system triggered the comparison between inferred *retrieved/assembly* location and correct location (end of *grasp* and *place phase*). While the *correctly identified samples* as the number of times the samples were labelled with the correct step number. Accuracy for *Intent detection* has been calculated by defining the *correctly identified samples* as any correctly labelled output of the Intent detection at any timestep of time t while *total samples* as the number of timesteps t of the dataset.

The best results for *Task tracking* were achieved using *Hand* (fig 5a) while best results for intention recognition were achieved by *Hand+Gaze* (fig 5b). *Error detection* performances were also evaluated using with *Hand* or G*aze* and their combination as well as the best combination of the best performing *Task tracking (Hand)* and *Intent detection (Hand+Gaze)* fig 5c. For both *task tracking* (Figure 4a) and *Intention recognition* (Figure 4b) model performance was assessed with an accuracy proportional score.

6.2 Step, Intention and Error Recognition

6.2.1 Task tracking results shows that hand signal was the most reliable signal for step recognition (99% accuracy). The results also show that, to an extent, the gaze signal on its own can be used to keep track of the manual procedure (72% accuracy). Additionally, the effectiveness of the *Gaze* signal seemed to be affected by both independent variables, as accuracy decreased with increased task complexity (number of blocks) and task uncertainty (presence of instruction). When both signals (eye + hand) were used, accuracy for step recognition was 97%.

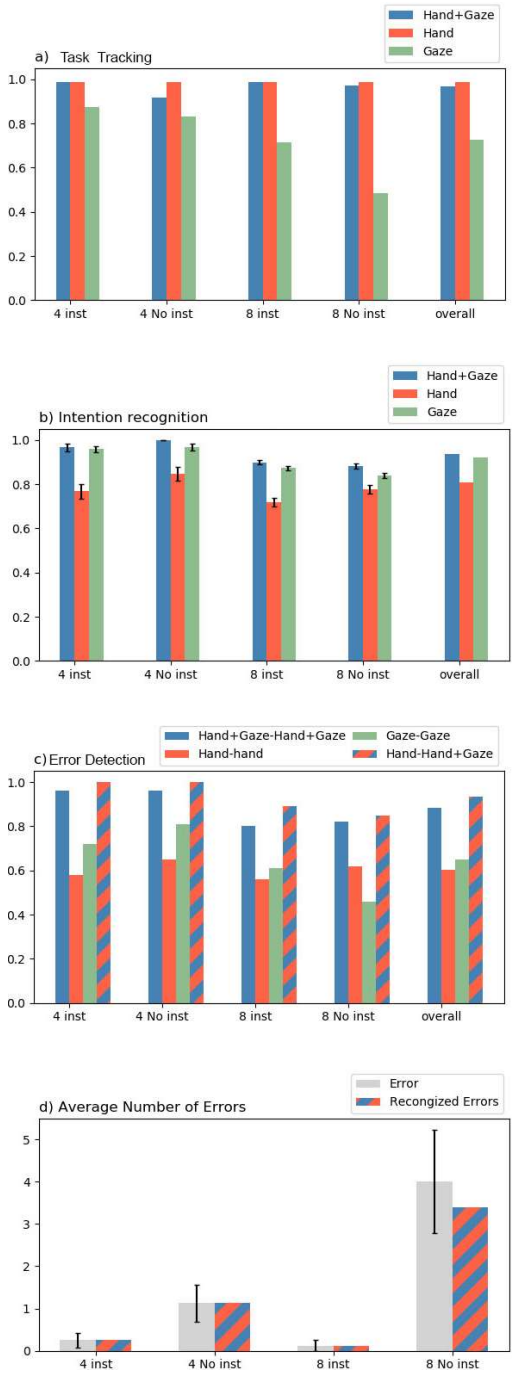

Figure 4 – Results. For each plot, the x-axis is the configuration of the experiment independent variable. Each of column is accuracy using *Hand, Gaze* or *Hand+Gaze.* Figure d displays a grey bar plot of the participants aggregated "Average Number of Errors per procedure", for each combination of the independent variables (complexity and instructions). While the stripped red-blue bars describe the number of errors recognized by the system using the combination of signals (*Hand, Hand+Gaze*). Error bars represent the standard error deviation across participants.

6.2.2 Intent detection results show that best detection performance was achieved pooling both signals (*Hand+Gaze*), with a 93.5% accuracy. Interestingly, in the case of user intent the worst detection performance was observed when the hand signal was used with 75% accuracy while gaze signal related performances were closer to the best performance with an overall 91% accuracy. Similar to step recognition, it is interesting to notice how the experimental independent variable complexity (number of blocks) yielded highest performance scores.

6.2.3 Error prediction: relied on comparing both results of step recognition and intention recognition. Since error detection relies on both step and intention recognition, different signal combinations were tested with the following logic: a configuration for each signal combination (*Hand, Gaze, Hand+Gaze*) and a configuration with the best performing signal from both step and intention (*Hand + Hand-Gaze*). To asses performances for this component we choose an F1 score [19].

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The best performing error detection results came from using *Hand* signal as step recognition and *Hand + Gaze* as intention recognition ~97%. Interestingly, *Hand* model ~59% performed worse than the *Gaze* model ~62%.

The average prediction time (i.e. duration between the prediction and the occurrence of the predicted event) is of 300ms. Prediction time is calculated from the moment the error comparison is triggered (i.e. confidence of the intention prediction or at the start of the detected gr or rl , see paragraph 4.1.4) to the end of the assembly or retrieve phase (i.e. detected end of gr or rl).

6.2.4 Behavioural results: Whenever a participant selected a building block from the resource area in an order that differed from the predefined sequence, the behaviour was labelled as an error. Accuracy (number of errors) scores were submitted to a 2x2 Repeated Measures ANOVA, with factors: Complexity (8 blocks vs. 4 blocks), Instructions (with vs. without instructions) (Figure 4d). A 2X2 Mixed ANOVA was performed on the dependent variable, with factors: Complexity (8 blocks vs. 4 blocks), Instructions (with vs. without instructions). The analysis revealed a main effect of Instruction ($F(1, 7)=8.452$, $p= 0.02$), a main effect of Complexity ($F(1, 7)=6.886$, $p= 0.034$), and an Instruction x Complexity interaction of ($F(1, 7)=8.690$, $p= 0.021$). As evidenced in Fig 4d, costs in performance induced by manipulations of task difficulty (Complexity and instruction availability) were reliably captured by the error detection algorithm.

7 Discussion

7.1 Error prediction

Error prediction relies on comparing the currently expected procedural step with the step the user intends to perform. Previous studies suggest that the richer the set of signals, the more reliable the intent prediction is [22]. Our findings are consistent with this:

we find best intent recognition performance is achieved with *Hand+Gaze* 93.5% followed by single *Gaze* signal 91% and then *Hand* signal 75%. However, the same conclusion does not apply to step recognition; the accuracy for *Hand* is 99% whereas for *Hand+Gaze* it is 97%. This can be explained by hand position being a closer representation of assembly states, whereas eye position reflects internal state such focus of attention and action intention [4, 5, 14].

Results shows that best performance of error detection are obtained when using *Hand+Gaze*, yielding a ~97% error detection accuracy ~300ms prior action completion (*retrieve/assembly*). Just using *Gaze* reduces the accuracy to 65% approximately 100ms in advance of action completion. Although this is much worse, it is still significantly better than chance performance of 12.5%.

7.2 Applications and use case scenarios

This system could be highly valuable in the context of factory assembly work, for example in scenarios where operators are required to accurately assemble parts in a specific sequence. Furthermore, it could be used in conjunction with state-of-the-art AR HMDs such as (HoloLens2). Hololens2 provide "out of the box" behavioural signals of Head, Eye-gaze as well as Hand position (tracking of both hand poses) which are required by our system to predict errors. Additionally, the headset's simultaneous localization and mapping (SLAM) enable to locate such signals onto a detected space (i.e. workspace) and therefore keep track of the relative position of the procedure station. AR HMDs could support the operator during the assembly process by displaying spatially-aligned-to-the-task instructions.

Our proposed architecture could drive an intelligent assistive system that by observing user behaviour (hand, eye-gaze) controls the displaying of AR instructions. For example, the automatic procedure tracking could be used to forward AR instructions as soon as a user complete a step (i.e. end of the step detected by the system). User intent and error prediction could trigger specific interventions aimed at mitigating error effects (i.e. such as displaying warnings or further instructions).

This intelligent assistive system could be used in scenarios such as operators training. Operator training (i.e. learning of new assembly procedures in industrial settings) is carried out using a set of instructions, with the supervision of an experienced operator [7, 8]. The proposed error detection system could work by providing automatic supervision in relation to the correct/erroneous execution of the procedure. For example, previous studies [7, 8] have highlighted how during the training process of new operators AR/superimposed instructions should be displayed up to a certain learning stage, and then removed as operators become acquainted with the procedure. However, while these studies do not offer insight about how and when to display or remove instructions, the concept of *faded scaffolding* [12] could be used. *Faded scaffolding* is a technique used in education to help beginners in acquire new skill while keeping them in their proximal zone of development (learning zone) [13]. Such concept would allow the operator to be in control of what information "fade" by essentially enabling him to turn on/off specific levels of instructions. Furthermore, our error detection system could extend the *Faded Scaffolding* by enabling a multimodal control for the display of the instruction, where both the user and the system can turn instructions on/off. Therefore, the proposed multimodal control would allow the operator to turn on/off the instructions (i.e. when instruction are perceived as overwhelming) but at the same time the system can trigger specific augmentations (i.e. display of instructions) when a user make a mistake (i.e. when the system detects an error). A predicted benefit for such system would be that it may require none or lower level of human supervisor during the training process.

On a final note, the proposed signals set, could be further tailor-suited signals available to specific headset units: for example, while some headsets (HoloLens2 and Magic leap) offer eye tracking natively, others do not (HoloLens). Given however that head position and eye gaze covariate within the context of manual tasks [4, 5, 14], head posture (position and orientation) could be a valid substitute for gaze directional signals.

7.3 Limitations and future directions

Our dataset does not cover incomplete steps as we ruled them out experimentally, (i.e. user grasping an object from a resource area and place it back immediately), however within assembly procedures this might happen and therefore future work should be addressing this behaviour.

Manual procedures are often characterized by a subset of steps that can be performed in variable order (e.g. when preparing a cup of tea, you might add sugar in the cup before or after pouring water in the cup, however you must boil water before you pour water in the cup). With our approach such flexibility is not handled and one potentially correct sequence amongst alternatives, would still be identified as incorrect. The definition of the SPT could be extended to support for multiple predefined sequences with minor adjustment to the logic of the error detection component.

Another limitation potentially relates to constraints in workspace layout. By reducing the layout size, the resulting step segmentation and user intent identification might be harder to extract, given limitations in spatial resolution of behavioural signal capture devices. For example, certain assemblies have a very small workspace layout (i.e. assembly a clock mechanism) for this type of procedure the signals of hand and eyes may not reveal a great deal of information.

While the system architecture has been developed to predict errors using real-time data (for example by ruling out bidirectional LSTM) the system has not been tested with real time data from an AR headset (i.e. HoloLens). It would be possible to connect the system to an AR headset via network (for example via web socket) in this case the computational cost of the RNNs would stay the same however costs related to the behavioural signals collections (i.e. hand posture tracking and eye tracking) and information transfer trough the network will have to be subtracted from the prediction time.

8 Conclusion

In this paper we presented a system architecture for error detection within SPTs. The challenge introduced by multi-stage

procedures consists in accurately segmenting the behavioural data into steps for which user intent can be compared to predefined action steps. To our knowledge there are no examples of a system that tackles real time error detection in the context of SPTs.

We present an automated *Eye-Hand* action identification and error detection system in the context of complex, order-constrained manual assembly procedures. Using *Hand* and G*aze* position, the system can detect assembly errors with 97% accuracy.

As part of the study we also evaluate different combinations of behavioural signals as inputs for the error detection system. We show an error detection accuracy of ~65% when eye-gaze signal is used alone, underling that eye-gaze can offer a substantially accurate depiction of procedure/step completion especially when the procedure complexity is minimal (accuracy of ~75% when the independent variable complexity is low). While overall results agree with literature suggesting that larger sets of signals provide best the handles over procedure representation, we found this untrue for the performance of specific system components: the subtask recognition component shows a drop-in performance when eye-gaze signal is added to the input signal set.

We extensively explored the tradeoffs between different combinations of measurements. We evaluated to what extent error detection performance varied as a function of combinations of behavioral signal inputs for the system's *task tracking* and *intention detection* components (*Hand+Gaze-Hand+Gaze, Hand-Hand, Gaze-Gaze, Hand-Hand+Gaze*,). We ultimately found that *Gaze* negatively impacted *step recognition* but increased the performance of *intention recognition*.

Finally, we focused on signals that are natively supported by current generation AR HMD units. Such units (i.e. Hololens2) offer all ingredients (eye-tracking and hand pose, and head pose relative to the space) required to implement our error detection system. Such integration could allow for an increased assistance encompassing "just in time" mitigation (i.e. alerts on predicted errors) as well as smart interface behaviour (i.e. assembly instructions display based on step recognition).

AKNOWLDGEMENTS

This study was supported by the HuMan project under the European Union's Horizon 2020 research and innovation programme, grant agreement no. 723737.
(http://humanmanufacturing.eu/).

REFERENCES

[1] Admoni, H. and Srinivasa, S. 2016. Predicting user intent through eye gaze for shared autonomy. *AAAI Fall Symposium - Technical Report*. FS-16-01-, (2016), 298–303. DOI:https://doi.org/10.1177/0278364913478447.

[2] Arthur Tang, Charles Owen, Frank Biocca, W.M. 2003. Comparative Effectiveness of Augmented Reality in Object Assembly. *CHI*. (2003).

[3] Bader, S. and Aehnelt, M. 2014. Tracking Assembly Processes and Providing Assistance in Smart Factories. May 2014 (2014), 161–168. DOI:https://doi.org/10.5220/0004822701610168.

[4] Biguer, B. et al. 1982. The coordination of eye, head, and arm movements during reaching at a single visual target. *Experimental Brain Research*. 46, 2 (May 1982), 301–304. DOI:https://doi.org/10.1007/BF00237188.

[5] Epelboim, J. et al. 1995. *The Function of Visual Search and Memory in Sequential Looking Tasks*.

[6] Funk, M. et al. 2015. A benchmark for interactive augmented reality instructions for assembly tasks. *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia - MUM '15*. Mum (2015), 253–257. DOI:https://doi.org/10.1145/2836041.2836067.

[7] Funk, M. et al. 2015. Stop helping me - I'm bored! Why assembly assistance needs to be adaptive. *UbiComp and ISWC 2015 - Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the Proceedings of the 2015 ACM International Symposium on Wearable Computers*. (2015), 1269–1278. DOI:https://doi.org/10.1145/2800835.2807942.

[8] Funk, M. et al. 2017. Working with augmented reality? A long-term analysis of in-situ instructions at the assembly workplace. *ACM International Conference Proceeding Series*. Part F1285, (2017), 222–229. DOI:https://doi.org/10.1145/3056540.3056548.

[9] González-Díaz, I. et al. 2018. Perceptually-guided Understanding of Egocentric Video Content. *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval - ICMR '18* (New York, New York, USA, 2018), 434–441.

[10] Huang, C.-M. et al. 2015. Using gaze patterns to predict task intent in collaboration. *Frontiers in Psychology*. 6, (Jul. 2015), 1049. DOI:https://doi.org/10.3389/fpsyg.2015.01049.

[11] Huang, C.-M. and Mutlu, B. 2016. *Anticipatory Robot Control for Efficient Human-Robot Collaboration*.

[12] Jackson, S.L. *The Design of Guided Learner-Adaptable Scaffolding in Interactive Learning Environments*.

[13] L. S. Vygotsky 1979. *Mind in Society: The Development of Higher Psychological Processes*.

[14] Pelz, J. et al. 2001. The coordination of eye, head, and hand movements in a natural task. *Experimental Brain Research*. 139, 3 (Aug. 2001), 266–277. DOI:https://doi.org/10.1007/s002210100745.

[15] Pimminger, S. et al. Low-Cost Tracking of Assembly Tasks in Industrial Environments. 86–93.

[16] Ratwani, R.M. and Trafton, J.G. 2011. A Real-Time Eye Tracking System for Predicting and Preventing Postcompletion Errors. *Human-Computer Interaction*. 26, 3 (2011), 205–245. DOI:https://doi.org/10.1080/07370024.2011.601692.

[17] Ravichandar, H. et al. 2016. Bayesian Human Intention Inference Through Multiple Model Filtering with Gaze-based Priors. *2016 19th International Conference on Information Fusion (FUSION)*. (2016), 2296–2302.

[18] Ravichandar, H.C. et al. 2016. Learning and Predicting Sequential Tasks Using Recurrent Neural Networks and Multiple Model Filtering. *aaai.org*. (2016), 331–337.

[19] Rijsbergen, V. and J., C. 1979. *Information Retrieval. 2nd. Newton, MA.*

[20] Sakata, N. et al. 2006. Visual assist with a laser pointer and wearable display for remote collaboration. *CollabTech*. (2006), 66–71.

[21] Sakita, K. et al. 2005. Flexible cooperation between human and robot by interpreting human intention from gaze information. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* (2005), 846–851.

[22] Schydlo, P. et al. 2018. *Anticipation in Human-Robot Cooperation: A Recurrent Neural Network Approach for Multiple Action Sequences Prediction*.

[23] Stiefmeierl, T. et al. 2006. Event-Based Activity Tracking In Work Environments. *3rd International Forum on Applied Wearable Computing*. (2006), 1–10.

[24] Wei, P. et al. 2018. *Where and Why Are They Looking? Jointly Inferring Human Attention and Intentions in Complex Tasks*.