Investigating speaker gaze and pointing behaviour in human-computer interaction with the *mint.tools* collection

Spyros Kousidis **Casey Kennington** **David Schlangen**
Dialogue Systems Group / CITEC / SFB 673
Bielefeld University
spyros.kousidis@uni-bielefeld.de

Abstract

Can speaker gaze and speaker arm movements be used as a practical information source for naturalistic conversational human–computer interfaces? To investigate this question, we recorded (with eye tracking and motion capture) a corpus of interactions with a (wizarded) system. In this paper, we describe the recording, analysis infrastructure that we built for such studies, and analysis we performed on these data. We find that with some initial calibration, a "minimally invasive", stationary camera-based setting provides data of sufficient quality to support interaction.

1 Introduction

The availability of sensors such as Microsoft Kinect and (almost) affordable eye trackers bring new methods of naturalistic human-computer interaction within reach. Studying the possibilities of such methods requires building infrastructure for recording and analysing such data (Kousidis et al., 2012a). We present such an infrastructure—the *mint.tools* collection (see also (Kousidis et al., 2012b))[1]—and present results of a study we performed on whether speaker gaze and speaker arm movements can be turned into an information source for an interactive system.

2 The *mint.tools* Collection

The *mint.tools* collection comprises tools (and adaptations to existing tools) for recording and analysis of multimodal data. The recording architecture (Figure 1) is highly modular: each information source (sensor) runs on its own dedicated workstation and transmits its data via the local area network. In the setup described in this paper, we perform motion capture via Microsoft Kinect and head, eye and gaze tracking via Seeingmachines Facelab 5.[2] We have developed specialised plugins that connect these sensors to the central component in our architecture, *Instantreality*.[3] This is a VR environment we use for monitoring the recording process by visualising a reconstructed 3D scene in real-time. A logging component simultaneously streams the timestamped and integrated sensor data to disk, ensuring that all data are synchronised. The data format is a shallow XML representation of timed, typed events.

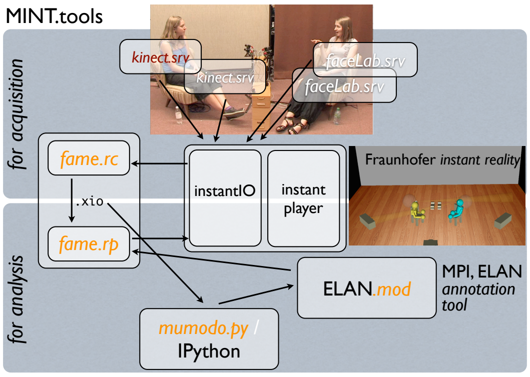

Figure 1: Overview of components of *mint.tools*; our contributions denoted by italics font. Top middle shows example lab setup; middle right shows corresponding VR scene, visualising motion capture and tracking of head posture, eye and gaze

The tracking equipment used in this setting is camera-based, providing for a minimally invasive setting, as subjects are not required to wear any equipment or tracking markers. In addition to the tracking sensors, video and audio are recorded us-

[1] Available at http://dsg-bielefeld.de/mint/.

[2] http://www.microsoft.com/en-us/kinectforwindows/, http://www.seeingmachines.com/product/facelab/, respectively

[3] Built by IGD Fraunhofer, http://www.instantreality.org

ing one HD camera. The AV channel is synchronised with the stream data from the sensors by means of a timecode in view of the camera.

Representative of the high modularity and flexibility of the *mint.tools* architecture is the ease with which components can be added. For the setting described here, a GUI was created which connects to the VR environment as an additional sensor, transmitting all of its state updates, which then are synchronously logged together with all other stream data from the trackers. This allows us to recreate the full scene (subject behaviour and the stimuli they received) in the virtual reality environment, for later inspection (see below Figure 6).

The analysis part of the *mint.tools* collection comprises a package for the Python programming language (described below) and a version of the ELAN annotation tool (Lausberg and Sloetjes, 2009), which we modified to control the replay of the virtual reality scene; this makes it possible to view video, annotations and the 3D reconstruction at the same time and in synchronisation.

Sensors are represented as nodes in a node-tree within the 3D environment. The values of data fields in these nodes are continuously updated as new data is received from the network. Using more than one sensor of the same type means simply another instantiation of that node type within the tree. In this way, our architecture facilitates tracking many people or complex setups where many sensors are required to cover an area.

3 Procedure / The TAKE Corpus

Our experiment is a Wizard-of-Oz scenario in which subjects (7 in total) were situated in front of a 40” screen displaying random Pentomino boards (Fernández et al., 2007). Each board configuration had exactly 15 Pentomino pieces of various colours and shapes, divided in four grids located near the four corners of the screen (see Figure 3 below). At the beginning of the session, a head and gaze model were created for the subject within the FaceLab software. Next, the subjects were asked to point (with their arm stretched) at the four corners and the center of the screen (with each hand), to calibrate to their pointing characteristics.

In the main task, subjects were asked to (silently) choose a piece and instruct the “system” to select it, using speech and/or pointing gestures. A wizard then selected the indicated piece, causing it to be highlighted. Upon approval by the subject, the wizard registered the result and a new board was created. We denote the time-span from the creation of a board to the acknowledgement by the subject that the correct piece was selected an *episode*. The wizard had the option to not immediately highlight the indicated piece, in order to elicit a more detailed description of the piece or a pointing gesture. What we were interested in learning from these data was whether speaker gaze and arm movements could be turned into signals that can support a model of situated language understanding. We focus here on the signal processing and analysis that was required; the model is described in (Kennington et al., 2013).

4 Analysis and Results

We perform the analyses described in this section using the analysis tools in the *mint.tools* collection, *mumodo.py*. This is a python package we have developed that interfaces our recorded stream data with powerful, freely available, scientific computing tools written in the Python programming language.[4] *mumodo.py* facilitates importing streamed data into user-friendly, easily manageable structures such as *dataframes* (tables with extended database functionality), or compatible formats such as Praat TextGrids (Boersma and Weenink, 2013) and ELAN tiers. In addition, *mumodo.py* can remote-control playback in ELAN and Instant Reality for the purpose of data viewing and annotation.

4.1 Gaze

Our post-processing and analysis of the gaze data focuses primarily on the detection of eye fixations in order to determine the pentomino pieces that the subjects look at while speaking. This knowledge is interesting from a reference resolution point of view. Although Koller et al (2012) explored listener gaze in that context, it is known that gaze patterns differ in interactions, depending on whether one speaks or listens (Jokinen et al., 2009).

Facelab provides a mapping between a person’s gaze vector and the screen, which yields an intersection point in pixel coordinates. However, due to limitations to the accuracy of the calibration procedure and noise in the data, it is pos-

[4] Especially IPython and Pandas, as collected for example in `https://www.enthought.com/products/epd/`. Example of finished analyses using this package can be found at `http://dsg-bielefeld.de/mint/mintgaze.html`

sible that the gaze vector does not intersect the model of the screen when the subject is looking at pieces near screen corners. For this reason, we first perform offline linear interpolation, artificially extending the screen by 200 pixels in each direction, by means of linear regression of the x, y components of the gaze vector with the x, y pixel coordinates, respectively ($R^2 > 0.95$ in all cases). Figure 2 shows the probability density function of intersection points before (left) and after this process (right), for one of the subjects. We see on the right plot that many intersection points fall outside the viewable screen area, denoted by the shaded rectangle.

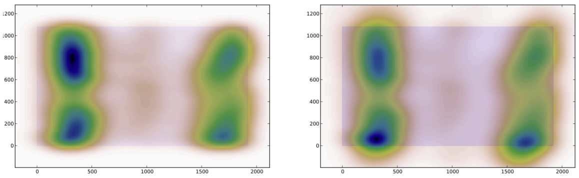

Figure 2: Probability density function of gaze intersections on screen before (left) and after interpolating for points 200 pixels around screen edges (right). Shaded rectangle shows screen size

In order to detect the eye fixations, we use two common algorithms, namely the I-DT and velocity algorithms, as described in (Nyström and Holmqvist, 2010). The I-DT algorithm requires the points to lie within a pre-defined “dispersion” area (see Figure 3), while the velocity algorithm requires the velocity to remain below a threshold. In both algorithms, a *minimum fixation time* threshold is also used, while a fixation centroid is calculated as the midpoint of all points in a fixation. Increasing the minimum fixation time threshold and decreasing the dispersion area or velocity (depending on the algorithm) results in fewer fixations being detected.

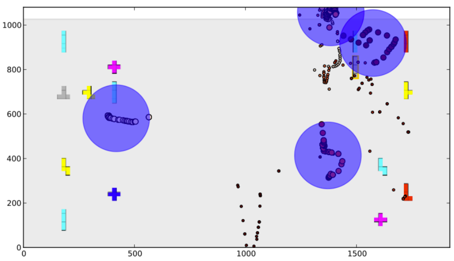

Figure 3: Fixation detection using the I-DT algorithm, circles show the dispersion radius threshold

Gaze fixations can be combined with information on the pentomino board in order to determine which piece is being looked at. To do this, we calculate the euclidean distance between each piece and the fixation centroid, and assign the piece a probability of being gazed at, which is inversely proportional to its distance from the centroid.

Figure 4 illustrates the gazing behaviour of the subjects during 1051 episodes: After an initial rapid scan of the whole screen (typically before they start speaking), subjects fixate on the piece they are going to describe (the “gold piece”). This is denoted by the rising number of fixations on the gold piece between seconds 5–10. At the same time, the *average rank* of the gold piece is higher (i.e. closer to 1, hence lower in the plot). Subsequently, the average rank drops as subjects tend to casually look around the screen for possible distractors (i.e. pieces that are identical or similar to the gold piece).

We conclude from this analysis that, especially around the onset of the utterance, gaze can provide a useful signal about intended referents.

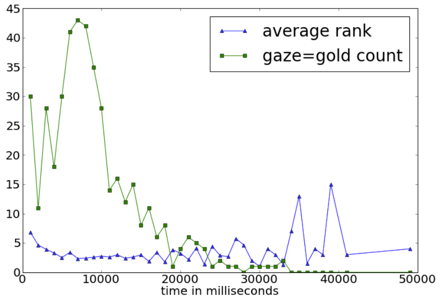

Figure 4: Average Rank and Counts over time (all episodes)

4.2 Pointing Gestures

We detect pointing gestures during which the arm is stretched from Kinect data (3D coordinates of 20 body joints) using two different methods. The first is based on the distance of the *hand* joint from the body (Sumi et al., 2010). We define the body as a plane, using the coordinates of the two *shoulders*, *shoulder-center* and *head* joints, and use a threshold beyond which a movement is considered a possible pointing gesture.

The second detection method uses the idea that, while the arm is stretched, the vectors defined by the *hand* and *elbow*, and *hand* and *shoulder* joints, respectively, should be parallel, i.e. have a dot product close to 1 (vectors are first normalised).

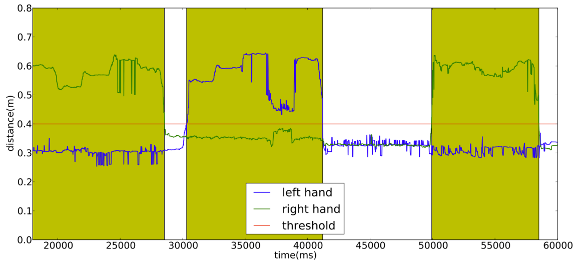

Figure 5: detection of pointing thresholds by distance of left(blue) or right(green) hand from body

In reality, the arm is never strictly a straight line, hence a threshold (0.95-0.98) is set, depending on the subject. The result of this process is an annotation tier of pointing gestures (for each hand), similar to the one shown in Figure 5. To make pointing gesture detection more robust, we only consider gestures identified by *both* methods, i.e. the intersection of the two annotation tiers.

Further, we want to map the pointing gestures to locations on the screen. Following a methodology similar to Pfeiffer (2010), we define two methods of determing pointing direction: (a) the extension of the arm, i.e. the shoulder-hand vector, and (b) the hand-head vector, which represents the subjective point-of-view (looking through the tip of one's finger). Figure 6 shows both vectors: depending on the subject and the target point, we have found that both of these vectors perform equally well, by considering the gaze intersection point (green dot on screen) and assuming that subjects are looking where they are pointing.

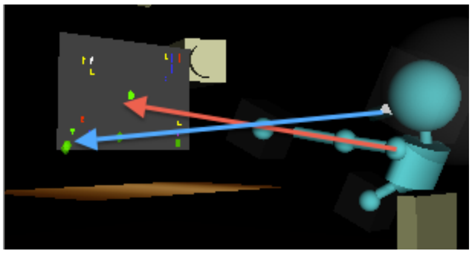

Figure 6: Hand-to-head and hand-to-shoulder pointing vectors

In order to map the pointing gestures to actual locations on the screen, we use the calibration points acquired at the beginning of the session, and plot their intersections to the screen plane, which we compute analytically, as we already have a spatial model of both the vector in question (Kinect data) and the screen location (Instantreality model).

Based on the pointing gestures we have detected, we look at the pointing behaviour of participants as a function of the presence of distractors. This knowledge can be used in designing system responses in a multimodal interactive enviroment or in training models to expect pointing gestures depending on the state of the scene. Figure 7 shows the result from 868 episodes (a subset that satisfies minor technical constraints). Overall, the subjects pointed in 60% of all episodes. Pieces on the board may share any of three properties: shape, colour, and location (being in the same corner on the screen). The left plot shows that subjects do not point more than normal when only one property is shared, regardless of how many such distractors are present, while they point increasingly more when pieces that share two or all three properties exist. The plot on the right shows that subjects point more when the number of same colour pieces increases (regardless of position and shape) and even more when identical pieces occur anywhere on the board. Interestingly, shape by itself does not appear to be considered a distractor by the subjects.

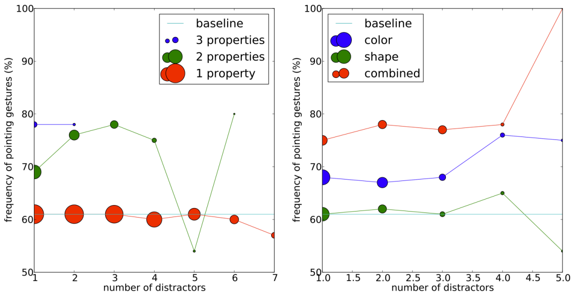

Figure 7: Frequency of pointing gestures as a function of the presence of distractors. Dot size denotes the confidence of each point, based on sample size

5 Conclusions

We have presented a detailed account of analysis procedures on multimodal data acquired from experiments in situated human-computer interaction. These analyses have been facilitated by *mint.tools*, our collection of software components for multimodal data acquisition, annotation and analysis and put to use in (Kennington et al., 2013). We will continue to further improve our approach for manageable and easily reproducible analysis.

References

Paul Boersma and David Weenink. 2013. Praat: doing phonetics by computer (version 5.3.48)[computer program]. retrieved may 1, 2013.

Raquel Fernández, Andrea Corradini, David Schlangen, and Manfred Stede. 2007. Towards Reducing and Managing Uncertainty in Spoken Dialogue Systems. In *Proceedings of the 7th International Workshop on Computational Semantics (IWCS'07)*, pages 1–3.

Kristiina Jokinen, Masafumi Nishida, and Seiichi Yamamoto. 2009. Eye-gaze experiments for conversation monitoring. In *Proceedings of the 3rd International Universal Communication Symposium*, pages 303–308. ACM.

Casey Kennington, Spyros Kousidis, and David Schlangen. 2013. Interpreting situated dialogue utterances: an update model that uses speech, gaze, and gesture information. In *Proceedings of SIGdial 2013*.

Alexander Koller, Maria Staudte, Konstantina Garoufi, and Matthew Crocker. 2012. Enhancing referential success by tracking hearer gaze. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 30–39. Association for Computational Linguistics.

Spyros Kousidis, Thies Pfeiffer, Zofia Malisz, Petra Wagner, and David Schlangen. 2012a. Evaluating a minimally invasive laboratory architecture for recording multimodal conversational data. In *Proc. of the Interdisciplinary Workshop on Feedback Behaviours in Dialogue*.

Spyros Kousidis, Thies Pfeiffer, and David Schlangen. 2012b. Mint.tools: Tools and adaptors supporting acquisition, annotation and analysis of multimodal corpora. In *to appear in Proc. of Interspeech 2013*.

Hedda Lausberg and Han Sloetjes. 2009. Coding gestural behavior with the neuroges-elan system. *Behavior research methods*, 41(3):841–849.

Marcus Nyström and Kenneth Holmqvist. 2010. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior research methods*, 42(1):188–204.

Thies Pfeiffer. 2010. *Understanding multimodal deixis with gaze and gesture in conversational interfaces*. Ph.D. thesis, Bielefeld University, Technical Faculty.

Yasuyuki Sumi, Masaharu Yano, and Toyoaki Nishida. 2010. Analysis environment of conversational structure with nonverbal multimodal data. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, page 44. ACM.