

Kinect Analysis: A System for Recording, Analysing and Sharing Multimodal Interaction Elicitation Studies

Michael Nebeling^{1*}, David Ott and Moira C. Norrie²

¹ Human-Computer Interaction Institute, Carnegie Mellon University

² Department of Computer Science, ETH Zurich
nebeling@cmu.edu, norrie@inf.ethz.ch

ABSTRACT

Recently, guessability studies have become a popular means among researchers to elicit user-defined interaction sets involving gesture, speech and multimodal input. However, tool support for capturing and analysing interaction proposals is lacking and the method itself is still evolving. This paper presents *Kinect Analysis*—a system designed for interaction elicitation studies with support for record-and-replay, visualisation and analysis based on Kinect’s depth, audio and video streams. Kinect Analysis enables **post-hoc analysis** during playback and **live analysis** with real-time feedback while recording. In particular, new visualisations such as skeletal joint traces and heatmaps can be superimposed for analysis and comparison of multiple recordings. It also introduces *KinectScript*—a simple scripting language to query recordings and automate analysis tasks based on skeleton, distance, audio and gesture scripts. The paper discusses Kinect Analysis both as a tool and a method that could enable researchers to more easily collect, study and share interaction proposals. Using data from a previous guessability study with 25 users, we show that Kinect Analysis in combination with KinectScript is useful and effective for a range of analysis tasks.

Author Keywords

multimodal interaction recording, visualisation and analysis; tools for guessability studies; Kinect Analysis.

ACM Classification Keywords

H.5.2. User Interfaces: Input devices and strategies; Evaluation/methodology.

INTRODUCTION

Recent guessability studies have produced a wide range of user-defined interactions including touch gestures for surface computing [14, 19], gestures for deformable displays and

* Michael Nebeling conducted this research while at ETH Zurich. He is now affiliated with Carnegie Mellon University and funded by a Swiss NSF Advanced Postdoc.Mobility grant, P300P2.154571.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

EICS'15, June 23 - 26, 2015, Duisburg, Germany
Copyright 2015 ACM 978-1-4503-3646-8/15/06\$15.00.
DOI: <http://dx.doi.org/10.1145/2774225.2774846>

mixed-reality environments [24, 27], motion gestures with mobile devices [21], as well as cross-device gestures [22, 23]. The principal method of showing the *effect* of a gesture and then prompting users for the *cause* by thinking aloud and demonstrating suitable gestures underlying most studies was popularised by Wobbrock et al. [28]. It has since been adapted for many different interaction modalities including full-body, speech-based and multimodal interactions. For example, Morris’s Web on the Wall study [12] collected user-defined interactions for web browsing in a living room TV setting around Kinect. However, rather than actually using Kinect, Wizard of Oz was employed and common browser functions demonstrated to elicit full-body gestures and speech commands that could have triggered them. Next to the challenges associated with collecting and analysing data using the method are challenges in applying the method itself to obtain unbiased interaction proposals [13] as well as issues in reproducing and sharing user-defined interaction sets [16].

This paper presents *Kinect Analysis*—a user interaction analysis system based on Kinect. The idea is that a Kinect sensor and Kinect Analysis are used for capture and analysis, without having to rely on video analysis or develop custom experiment software, which many previous studies required. Kinect Analysis was primarily designed for studies in the lab on Kinect-based systems [6, 8] and elicitation exercises [11, 12], but we also discuss how it could cater to other types of studies involving multiple users and different devices.

Our work offers three main contributions: **1) Kinect Analysis as a tool:** Our system uses multiple annotated timelines for analysing and comparing Kinect recordings. In the context of an analysis, user-defined parts of a recording can be manually annotated and labelled. Annotations can also be generated automatically by log file parsers and scripts. Two analysis modes are available: live analysis while recording and post-hoc analysis during playback. During recording or playback, joints can be displayed selectively so that joints that are not important for certain movements or gestures can be suppressed. Joint movements can also be visualised using traces and heatmaps. **2) KinectScript:** We introduce *KinectScript* for querying Kinect recordings using a combination of skeleton, distance, audio and gesture scripts and automating coding and analysis tasks. **3) Kinect Analysis as a method:** Based on these innovations, we explore the benefits and limitations of using Kinect Analysis. We focus on the evaluation of Kinect Browser [16], a new Kinect-based system developed based on Morris’s guessability study [12].

BACKGROUND AND REQUIREMENTS

Designing and conducting user studies around natural user interfaces remains a major challenge for two main reasons.

First, rather than traditional mouse and keyboard input that is easy to track, the body is used as input [10]. A system under evaluation is usually only able to track correctly performed gestures and speech commands. Recording all user interactions is required to detect commands that were intended, but not recognised. It is therefore either necessary to build special experiment software or record video.

Second, once the user interactions have been recorded, analysing the collected data poses another challenge. Video analysis is a common method involving transcription and annotation of video recordings [4]. Video data is very rich and contains a lot of information—often several passes through the data are required. Hence, manual annotation and reviewing of video sequences is time-consuming and cumbersome. Automated video analysis is difficult and only feasible for specific application areas and predefined user interactions.

The design of Kinect Analysis was driven in two ways. First, we conducted a systematic review of existing tools and literature on previous studies to analyse common requirements, analysis tasks and workflows. Second, we carried out Kinect-based and guessability studies as part of our own research.

To the best of our knowledge, there is no dedicated tool for analysing and comparing user interactions based on Kinect. *DejaVu* [9] is an IDE extension that enables programmers to easily record, review, and reprocess temporal data to iteratively improve the processing of Kinect camera input. *DejaVu* uses a canvas to visually and continuously monitor the inputs, intermediate results, and outputs of computer vision processing techniques. While useful to developers, it does not share our goal of supporting evaluators in the analysis process. *MAGIC* [1] is a motion gesture design tool that provides facilities for experimenting with motion gestures. A key feature of *MAGIC* is retrospection, allowing designers to review previous actions by visualising recorded gestures and making a video recording available. *GestureAnalyzer* [7] provides support for interactive hierarchical clustering of gesture data based on multiple-pose visualisations. It shares Kinect Analysis' vision to support researchers in performing elicitation studies, yet it is limited to gesture analysis tasks.

We designed Kinect Analysis as an extensible record-and-replay tool for gesture, speech and multimodal interactions. In addition to an extensible set of visualisations such as skeletal joint traces and heatmaps, it supports scripting of complex analysis tasks involving multimodal interactions in a declarative manner based on KinectScript. While we could not try out the tools listed above, we believe that they are complementary to, and quite powerful when used in combination with, Kinect Analysis. One limitation of *GestureAnalyzer*, for example, is that it requires a specific gesture data format, where gestures start and end with a natural standing pose. This seems to be quite limiting, especially for user-driven elicitation studies. To mitigate this, Kinect Analysis could be used for recording and initial coding. After segmentation

and annotation with Kinect Analysis, gesture data could be exported to *GestureAnalyzer* where it may be processed further for hierarchical clustering and identification of descriptive features in user-defined gesture sets.

Also related to Kinect Analysis are tools for qualitative video analysis. The conventional approach of video analysis consists of watching a video sequence with a media player while taking notes using a spreadsheet application or the like. *VACA* [2] integrates both video viewing and annotation into one system. It provides video timeline annotations that can also be imported from logs to improve the rate at which video analysis can be performed. *ChronoViz* [3] is a more general tool for navigating, visualising and inspecting multiple streams of time-coded data that recently added support for Kinect recordings [26]. Kinect Analysis introduces new visualisations and tools for automating analysis tasks based on Kinect's skeletal tracking and speech recognition, allowing evaluators to keep track of previously performed analysis tasks and share their Kinect data and analyses with others.

Studies we have taken into account range from gesture-based [5, 14, 19], to multi-device [22, 23], to Kinect-based studies [11, 12, 25]. We can observe a proliferation of guessability studies for different kinds of situations and settings. Common to most studies is that they employ the think-aloud protocol, often complemented by log files and video analysis, as suggested in the original study design of Wobbrock et al. [28]. Although most studies involve devices with different kinds of input tracking sensors, only a few actually made use of recognisers [16, 20] and many required the development of custom experiment software [15, 21]. Researchers are still experimenting with the method itself to overcome potential biases. A recent proposal is controlling and stimulating the production of interaction proposals, priming participants and using partners in elicitation exercises [13]. It is common to report user agreement scores and develop taxonomies often illustrated with sketches or scene stills of interaction proposals. However, so far little has been done to record and share data in a format more practical for direct comparison, reproduction and implementation. One exception is the previously conducted Kinect Browser study [16].

Kinect Browser is a Kinect-based multimodal web browser. Informed by Morris's Web on the Wall study [12], Kinect Browser implements 10 common browser functions and recognition code for 9 gestures and 16 speech commands. The study involved 25 participants and was divided into three tasks. Task 1 was essentially a guessability study based on Morris's study design. Wizard of Oz was employed to collect interactions proposals for the 10 browser functions. Task 2 was similar to Task 1, but instead of the Wizard, Kinect Browser was used and stepwise configured to react to preferred interactions. Finally, during Task 3, participants used their preferred interactions to browse a web site for planning a trip over the weekend. All three tasks were recorded and analysed using an earlier version of Kinect Analysis. Kinect Browser was extended to produce user logs for successfully recognised gestures and speech commands and the triggered browser functions. Task 3 was also recorded on camera.

The study had multiple goals: (1) it was interested in finding out about the proportion of gesture, speech and multimodal commands proposed for controlling the browser, (2) by dividing the elicitation exercise into two parts using Wizard of Oz in Task 1 and Kinect Browser in Task 2, it wanted to investigate the effects of using recognisers in guessability studies, (3) it wanted to investigate user agreement in terms of Morris’s consensus metrics [12] and compare preferred interactions to her study. Moreover, in the context of this paper, it serves as a running example and provided a basis for our comparison of macro-level video analysis to KinectScript-based analysis in terms of the process, efficiency and quality of results, which we report later. The researchers also wanted to release the code and data to allow the community to easily build on the results, which motivated means for sharing.

Based on this review and our own experience, we devised the following requirements for Kinect Analysis.

(R1) Timeline-based data navigation and annotation

Common to all analysis tools including ours is that they manage, process and visualise time-coded data as well as providing means for annotating the timeline. It is also possible to incorporate external data sources such as log files.

(R2) Integration and visualisation of Kinect data

Not all tools can record data, but most handle a variety of input streams and some added support for Kinect. Like DeJaVu, Kinect Analysis can record all Kinect data streams. It adds joint traces and heatmaps to typical skeleton visualisations.

(R3) Control over Kinect parameters

To accommodate different study settings, Kinect’s tilt angle, joint filtering, skeleton selection and tracking mode must be maintained. Skeleton selection is required for multi-user settings and the tracking mode for full-body/seated gestures.

(R4) Feedback on Kinect tracking quality

When recording Kinect, tracking quality is important. Kinect Analysis produces statistics on inferred vs. tracked joints. This gives live feedback to study facilitators and the statistics may also be correlated to participant feedback [16].

(R5) Organisation of recordings and analyses

Kinect Analysis supports tagging of entire recordings and organising them into multiple analyses, while results can be stored and retrieved later. Although being crucial for video analysis [4], retrospection was so far unique to MAGIC.

(R6) Detection of gestures and speech commands

Like DeJaVu, Kinect Analysis processes Kinect’s depth data to detect predefined gestures using Kinect’s default or custom recognisers. Kinect Analysis can also recognise predefined speech commands and mulimodal interactions.

(R7) Metrics for coding interaction proposals

Important for guessability studies is user agreement [28] and taking note of preferred interactions in the case of *multi-modal synonyms* [12]. Our KinectScript enables querying and declarative coding by skeleton, distance, gesture and speech.

(R8) Auto recording, tagging and labelling

Kinect Analysis can produce recordings on skeleton detection and also generate tags and labels when gesture, speech or multimodal commands are detected. This is useful for production phases in elicitation exercises [13].

(R9) Comparison of multiple recording sequences

Kinect Analysis supports simultaneous playback of two Kinect recordings and allows analysis tasks to be carried out across multiple recordings. This is again important for determining user agreement in guessability studies.

(R10) Sharing of recordings and reproduction of analyses

All data recorded and analysis tasks created with Kinect Analysis can be exported to external formats (MP4 video, MP3 audio, JSON Kinect data, CSV/PNG analysis data). They can be reimported or processed using external tools.

KINECT ANALYSIS

In this section, we describe the Kinect Analysis system and give more details on how its design caters for each requirement. Figure 1 shows the main components. Kinect Analysis is based on a client-server architecture. We decided to implement a web-based client so that Kinect Analysis can be accessed and controlled from any web-enabled device. Importantly, the client used for analysis can be different from the one used for recording, which gives more flexibility during evaluations and allows researchers to have shared access and easily exchange data. The client can be used to view existing recordings and analyses, create new ones and configure the aforementioned Kinect parameters. It also implements different visualisers which we detail below. The server is composed of four main components: replay system, Kinect data recorder, KinectScript processor and database. The server accesses gesture and speech recognisers and the Kinect itself. Our implementation uses Kinect Interactions and Windows system speech service as default recognisers, but, as will be discussed later, it is possible to provide custom recognisers.

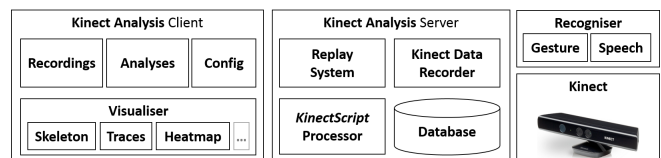


Figure 1. Architecture of Kinect Analysis system

The remainder of this section introduces the main features of Kinect Analysis and is structured into three parts: (1) recording and visualising multimodal interactions, (2) coding and comparing interaction sets, and (3) sharing data and analysis tasks. To illustrate the usage, we include statements in *italics* on how the features were used for the Kinect Browser study [16] introduced earlier.

Recording and visualising multimodal interactions

Kinect Analysis allows all Kinect data streams to be recorded. Kinect’s skeleton and audio streams are recorded per default. The skeleton stream tracks the location and direction of joints

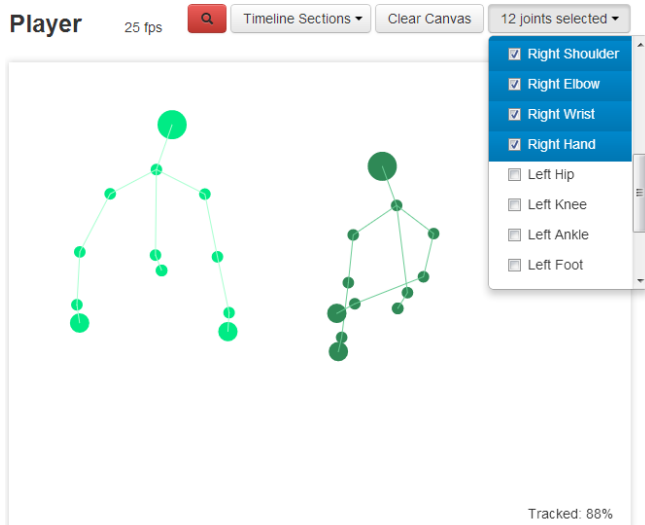


Figure 2. Player showing a recording containing two skeletons; lower body parts excluded via joint selection.

over time. If more details are required, recordings of Kinect’s depth, colour and infrared streams can be included.

For the Kinect Browser study, skeleton and audio streams were recorded with Kinect Analysis. Compared to video, this provides compact data that has the advantage that the privacy of recorded humans is preserved and no post-processing required before sharing the data in order to conceal faces or other sensitive data that would be visible on video.

Kinect Analysis supports two modes of recording new sequences: manual recording and auto recording. In manual recording mode, Kinect Analysis creates a new sequence and continuously records all selected data streams until a button to stop recording is pressed. All recorded data is then stored in one single recording. However, this can mean that a recording will be produced even if no skeleton was detected. In contrast, the auto mode only records as soon and as long as at least one skeleton is detected. When all skeletons leave the Kinect’s field of view, the current recording is finished and added to the stored recordings. The auto mode remains active and additional recordings will be created with a new skeleton being recognised. Auto recording prevents wasting recording time and storage space. It facilitates unsupervised studies where a Kinect device is deployed to capture interactions without a study facilitator being present. However, it also supports study facilitators in that they must not ensure for every participant that the device is in fact recording.

Since the Kinect Browser study involved three tasks collecting feedback through post-task questionnaires, a mix of manual recording between tasks and auto recording within tasks were used. For easy lookup, recordings were named with the participant ID and tagged according to the task of the study. In total, 23+23+23 recordings were created for the three tasks and 25 participants. Unfortunately, there was a technical problem with one recording, three times the study facilitator accidentally did not record the next task, one recording was started late and one was cancelled.

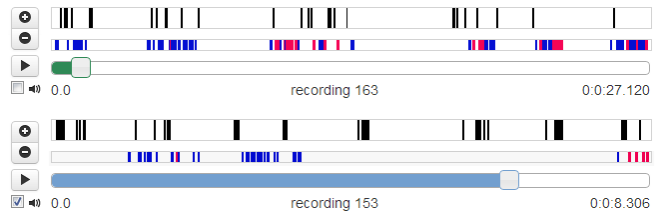


Figure 3. The interface contains controls and timelines for two recordings, here 163 (green) and 153 (blue). Playback of each recording can be controlled and timeline sequences annotated. Manual annotations (black) and detected gestures (blue) and speech commands (red) are coded on separate timelines grouped with each recording.

The player is clearly the heart of Kinect Analysis. It renders Kinect’s skeleton and video data and provides features for controlling playback, annotating recordings and performing analyses. It consists of a large canvas and several controls for visualisation (Figure 2) as well as a timeline interface (Figure 3). The canvas not only shows skeletal data, but also conveys tracking information. Besides colour coding of tracked and inferred joints, the percentage of tracked joints is displayed. The timeline sections dropdown above the canvas can be used to select one or more sequences for playback. This is practical for re-watching and in-depth analysis. Moreover, joint selection is useful to focus on certain body parts.

The Kinect Browser study was conducted in a controlled environment with seated participants. Seated tracking mode and only joints pertaining to the upper body were selected.

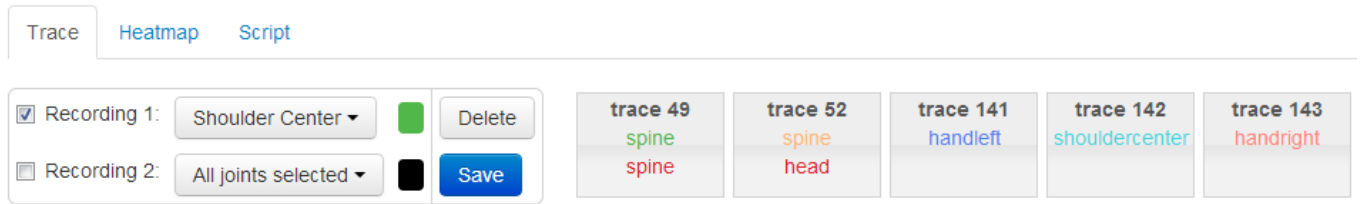
Coding and comparing interaction sets

Kinect Analysis supports two modes of analysis: live analysis and post-hoc analysis. Both are conducted in realtime. Live analysis creates visualisations or timeline annotations while recording, but is limited to the current recording. Post-hoc analysis does so during playback of existing recordings, and can involve multiple sequences from different recordings.

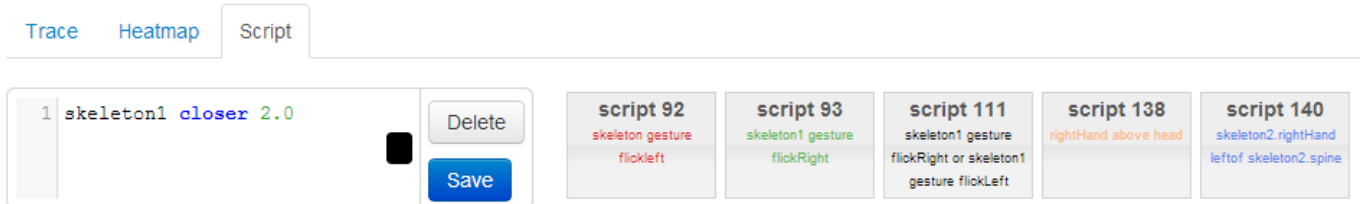
For the Kinect Browser study, a mix of live analysis for auto coding the predefined 9 gestures and 16 speech commands and post-hoc analysis for all interaction proposals that were not detected by the system were used. Below are some results.

The timeline interface shown in Figure 3 not only indicates playback progress and supports seeking through recordings, but also allows timeline sequences to be annotated. As a common requirement, manual annotation of sequences of arbitrary length is supported by the system. Tags can be added to label the timeline sections. Automatic annotation is supported in two ways. First, it is possible to import log files created by external software. Second, Kinect Analysis can detect predefined gestures and speech commands. We will explain later how new commands can be added to the system.

Figure 3 shows an example from the Kinect Browser study where flick hand gestures and “back” or “forward” speech commands are marked in different colours. Here, the timeline interface gives a first visual impression of the gesture/speech ratio which can be interesting for analysis of multimodal synonyms [12]. Manual annotations were created for interaction proposals that could not be detected by the recognisers.



(a) A trace configuration can include one or two recordings and a selected number of joints. A colour picker is provided to choose a trace colour for each recording. The right side shows a list of defined trace configurations.



(b) Script editor for writing new scripts using our KinectScript scripting language. The colour picker defines the colour of timeline annotations generated by the script. The right side shows a list of defined scripts, e.g. skeleton gesture flickleft for marking flick hands gestures in red.

Figure 4. The analysis toolbar is used for managing trace, heatmap and script configurations that can be applied to analyse recordings.

Kinect Analysis implements three different analysis tasks: traces, heatmaps and scripts written in *KinectScript*—our scripting language for querying recordings and declaring custom analysis tasks as detailed in the next section. Traces and heatmaps were implemented as two visualisation techniques for analysing joint movements. Traces depict the location history of one or more data points over time. On the other hand, heatmaps are commonly used for visualising three dimensional data—two dimensions represent x and y coordinates and the third dimension is used for showing the frequency of a data point in relative comparison to the absolute minimum and maximum of the dataset. In Kinect Analysis, data points are coordinates of selected joints. To indicate the frequency in heatmaps, red (hot) is used for maxima and blue (cold) for minima. Thus, areas that were more frequently covered appear more red and less frequently covered areas more blue.

For the Kinect Browser study, traces and heatmaps were created at a per-gesture level for selected participants to analyse movements and the role of hands. In addition, scripts were defined for coding gestures, speech commands and multimodal interactions. See the video and supplementary material accompanying the paper for more details on this.

Sharing data and analysis tasks

Heatmaps, traces and scripts can be configured in a toolbar. For example, traces can be configured to include selected joints of one or two recordings (Figure 4(a)). A different colour can be associated with each recording so that the traces originating from the different recordings can be distinguished based on colour-coding. While traces and heatmaps generate essentially an image, scripts generate timeline annotations that indicate when a certain event occurred. The analysis details view combines associated recordings and generated results in collapsible sections (Figure 5). For comparison, multiple traces and heatmaps can be superimposed on the player canvas. Timeline annotations of two recordings can be examined via the timeline interface (Figure 3).

Analysis 14







Recordings			
Traces			
Heatmaps			
ID	Heatmap ID	Recording ID	Thumbnail
35	44	99	 ✕
34	9	99	 ✕
33	44	98	 ✕
32	9	98	 ✕
31	44	97	 ✕
30	9	97	 ✕
Script Outputs			

Figure 5. Heatmaps section with thumbnail previews of previously created heatmaps. Similar previews are available for traces, while previews of script outputs show colour-coded timeline annotations.

The main analyses of the Kinect Browser study required 2 trace configurations and 2 heatmap configurations for tracking movements of the left and right hand, respectively. Additional scripts were defined for coding the 9 gestures, 16 speech commands, and 2 multimodal interactions involving both gesture and speech. These scripts allowed the researchers to automatically annotate a good number of interaction proposals. However, due to some limitations discussed later, manual annotation was still required to catch all suggested multimodal interactions and label them properly.

Script Type	Structure	Example
Skeleton Script	skeletonIdentifier.joint positionKeyword skeletonIdentifier.joint	handright above head skeleton1.handleft above skeleton1.spine
Distance Script	skeletonIdentifier distanceKeyword value	skeleton1 closer 1.0 skeletonLeft closer 2.0 skeleton1.handleft closer 2.0
	skeletonIdentifier skeletonIdentifier distanceKeyword value	skeleton1 skeleton2 closer 2.0 recording1.skeleton1 recording2.skeleton1 farther 1.0
Audio Script	skeletonIdentifier says speechTerms	skeleton1 says next skeleton1 says back,go back
Gesture Script	skeletonIdentifier gesture gestureName	skeleton1 gesture flickleft skeleton1 gesture flickright
Combined Script	script1 and script 2 script1 or script2	skeleton1 says next and handright above head skeleton1 says next or skeleton1 gesture flickleft

Figure 6. Overview of KinectScript with examples on how to construct scripts of a certain type

recordingIdentifier	positionKeyword	joint	HandRight	ShoulderLeft
recording1	above	AnkleLeft	Head	ShoulderRight
recording2	below	AnkleRight	HipCenter	Spine
skeletonIdentifier	leftof	ElbowLeft	HipLeft	WristLeft
skeleton1	rightof	ElbowRight	HipRight	WristRight
skeleton2	distanceKeyword	FootLeft	KneeLeft	gestureName
skeletonLeft	closer	FootRight	KneeRight	grip
skeletonRight	farther	HandLeft	ShoulderCenter	push

Figure 7. Supported keywords in KinectScript

A key feature of Kinect Analysis for sharing is that it manages recordings and analyses separately. The benefit is that the same recording can be subject of multiple analyses and different analysis tasks and generated outputs can be stored and retrieved together with each analysis. For example, the same recording can have different annotations when it occurs in different analyses. Configurations are globally defined via the toolbar and can thus be reused not only between analysis tasks pertaining to the same study, but even between studies.

For each of the 25 participants in the Kinect Browser study, a new analysis was created and associated with the recordings from each task. For selected pairs of participants, additional analyses using traces and heatmaps were created post-hoc involving only Task 2 and Task 3 recordings.

IMPLEMENTATION

The client is implemented using HTML5, CSS3 and JavaScript in combination with the jQuery library. The server is implemented in C# using a MySQL database for storage and retrieval of data streams and the Kinect SDK 1.8 to interact with the Kinect sensor. Client and server communicate via the WebSocket protocol and exchange data using JSON.

KINECTSCRIPT

The main idea behind KinectScript is to provide means for querying recordings based on scripting expressions that may evaluate to true and annotate the timeline depending on conditions or events that occur. We designed KinectScript to be simple, modular and extensible. As a proof of concept, we focused on exploring different types of scripts and their potential, rather than developing a full-fledged language.

Our KinectScript implementation supports scripts of four different types (Figure 6): **1) Skeleton scripts** consider relative positions of joints. Using skeleton and joint identifiers (Figure 7), a relation between two joints can be formulated

and will be evaluated with respect to the current joint positions. **2) Distance scripts** can be used to measure the distance from skeletons to the Kinect or between skeletons. **3) Audio scripts** search for expressions in recorded speech. **4) Gesture scripts** can be used to recognise predefined gestures.

KinectScript also supports scripts that involve two recordings. Scripts can be combined using conjunction *and* or disjunction *or*. Other logical connectives such as negation *not* and parantheses to control order of precedence could be added in the future by extending the language and our implementation.

Skeleton Scripts

Skeleton scripts define relations between joints. An example for a skeleton script is `leftHand above head`. It evaluates to true if the left hand of a skeleton is above the head. As no skeleton identifier is specified, all skeletons in a recording will be considered. Two different skeleton identifiers are implemented so far. Skeletons can be identified by tracking ID (e.g. `skeleton1` for the first tracked skeleton) or by position (e.g. `skeletonLeft` for the skeleton left of another skeleton). Others (e.g. by distance or by colour) could be added later.

Distance Scripts

Two subtypes of distance scripts are supported. The first evaluates the distance between a joint of a skeleton and the Kinect sensor. The spine is used if no joint is specified. Distance values are in metres. For example, the script `skeleton1 closer 1.0` returns true if the distance between the first skeleton's spine and the Kinect is less than one meter. The second subtype evaluates the distance between the spines of two skeletons. The skeletons can either be in the same recording or in two different recordings. For example, using the `|` operator as in `recording1.skeleton1|recording2.skeleton1 farther 1.0` compares the distance between the first skeleton in a recording on timeline 1 and the first skeleton in a recording on timeline 2, and returns true if it is less than 2 metres. New subtypes could be added in the future, e.g. distance between two joints or between skeletons and a fixed reference point.

Audio Scripts

Audio scripts search for recognised terms within recordings. As an example, the script `skeleton1 says back,forward` searches for the terms "back" or "forward". Alternative terms are separated by comma and terms composed of multiple

Click Link	dwell	click/grip	push/press	othergesture	"link #"	"link title"	multimodal	Gesture	Speech
Total Task 1	3	15	11	6	1	8	14	35	9
Total Task 2	7	23	9	7	0	5	5	46	5
Preferred 1	0	11	5	1	0	3	7	17	3
Preferred 2	0	18	1	2	0	3	5	21	3

Figure 8. Kinect Analysis was used in combination with Excel spreadsheets for scoring interaction proposals, here for the Click Link referent.

words by space, e.g. “go back”. Currently, skeleton identifiers are ignored in audio scripts, as speech is recorded independent of skeletons. Given Kinect’s microphone array that can track the direction of sound, it is in principle possible to correlate the spatial sound information with the physical position of a skeleton to match only for the specified skeleton.

Gesture Scripts

Gesture scripts test for a certain gesture and evaluate to true if the specified gesture was detected. The `gesture` operator is followed by a keyword reference to a predefined gesture recognition function. Using Kinect Interactions, Kinect Analysis can detect `grip` and `press` hand gestures per default. An example of a custom gesture script used in the Kinect Browser study is `skeleton1 gesture flickright` for detecting flick hands gestures from left to right. Custom recognition code can be supplied to the Kinect Analysis server. Gesture functions are expected to process Kinect’s data streams and return true if the gesture was successfully recognised. Gesture functions may also return a timespan for marking the beginning and the end of the gesture if it can be segmented.

Script Evaluation

The general method behind script evaluation is to parse a script and then inspect every single skeleton frame on the Kinect Analysis server and annotate the timeline position on the client if the script condition evaluates to true. For example, in the case of a custom `flickright` gesture script, the client will add a marker to the timeline (blue) and label it with the gesture keyword “flickright”. For gesture functions returning a timespan, the respective timeline sequence will be marked in retrospect. This principle applies to all but audio scripts.

Audio scripts are evaluated as follows. In a first step, a grammar is constructed from the terms contained in the audio script. In a second step, the Windows system service is used to perform asynchronous speech recognition on the WAV file associated with the selected recording, returning position and confidence levels of the detected words. Once speech recognition is finished, a list of recognised speech objects is serialised to JSON and sent to the client where the results are marked on the timeline. Hence, audio script evaluation does not require replaying the entire recording. Moreover, audio scripts are only evaluated once and then the result is cached.

The same principle also applies when scripts are combined, e.g. audio and gesture scripts as in `skeleton1 says next and skeleton1 gesture flickleft`. For the combined script to evaluate to true, the flick hands gesture has to occur within the range of an audio event. Speech recognition detects the term “next” at position x in the recording. Currently, true is returned if the flick right gesture was detected within the range of $x \pm \delta$ where δ is a globally configurable threshold of 2 seconds per default. Future versions of KinectScript could make

such time constraints configurable at a per-script level by adding new operators, e.g. `skeleton1 says next and right-Hand above head within 3s`, for a 3 seconds threshold.

KINECT BROWSER STUDY

Expanding on previous examples from the Kinect Browser study used in the paper, this section presents in detail how Kinect Analysis was used for capture and analysis. Note that our intention here is to illustrate the use of Kinect Analysis and the role it played in the Kinect Browser study. Readers interested in the actual results of the study are referred to [16].

Using a dual-monitor setup, a single computer and Kinect sensor were used for running Kinect Browser and Kinect Analysis in parallel. The participant’s screen was a ca. 63” projection surface with 1024x768 resolution showing Kinect Browser. All interactions as they were tracked by Kinect were recorded and previewed on the study facilitator’s screen. For Task 3, a second projector was used to display goals and six sub-tasks guiding participants through the trip planning scenario. Using the same computer made sure that timestamps in the user logs generated by Kinect Browser matched those of the Kinect recordings created using Kinect Analysis.

In preparation for the analysis, Kinect Analysis was customised with gesture and speech recognition code from the Kinect Browser system. The analysis was conducted in three steps. First, the researchers sequenced Task 1 and Task 2 recordings by the 10 referents, and Task 3 recordings by the six sub-tasks, and coded each at a per-interaction level. Second, CSV data export from Kinect Analysis to Excel spreadsheets was used for counting interactions and scoring user agreement according to [12]. Third, an in-depth analysis of selected recordings made use of specific features of Kinect Analysis to analyse the use of gesture and speech.

In total, 907 interactions almost equally split between Tasks 1 and 2 were coded. Both tasks involved a production phase in which participants were prompted for possible gesture, speech and multimodal commands as well as their overall preference for each referent. Figure 8 shows an excerpt for the Click Link referent with the number of total and preferred interactions broken down by modality. The complete analysis revealed that participants most commonly suggested gesture (Task 1: 65%, Task2: 60%), then speech (Task 1: 31%, Task 2: 39%), and rarely multimodal interactions composed of both gestures and speech commands (Task 1: 4%, Task 2: 1%). In Task 3, participants used 84% gesture and 16% speech for 662 successfully registered browser function calls.

Use of Traces and Heatmaps

In-depth analysis of selected recordings making use of traces and heatmaps allowed the researchers to make several interesting observations [16]. First, creating heatmaps of hands

ID	Length Kinect Recording	Length Video Recording	Tracking Ratio
P8	0:10:23	0:09:31	0.88
P11	0:04:04	0:05:18	0.95
P16	0:09:57	0:11:21	0.68
P22	0:03:56	0:05:34	0.74

Table 1. Selected Kinect and video recordings for comparison

at a per-participant level, the researchers investigated the use of the dominant hand and how it varied between left and right-handed participants. Moreover, hand traces created per task showed a generally smaller interaction window for Task 3 where participants attempted to perform gestures more efficiently. Breaking the traces further down by referent, the researchers found that participants assigned different roles to hands when switching between referents and depending on which side of the screen required interaction. For those participants who experienced poor gesture recognition, per-gesture traces showed for several attempts that the intended gesture, in particular, the flick hand gesture for the Go Back/Forward and the Switch Tab referents, would have been recognised had they performed a larger hand movement. At the same time, the analysis revealed some potential for relaxing the gesture recognition code by allowing for a smaller interaction window without necessarily raising the potential for conflicts with other gestures. Finally, heatmaps alarmed the researchers that participants accidentally triggered interactions using grip gestures while resting their hands on the lap or using the armrest without paying attention to that hand.

KinectScript vs. Video Analysis

In the next step, we compared KinectScript against video analysis using data from Task 3. The idea here was not to try to show that one method might be better than the other, but to compare the process, efficiency and quality of results. We first explain the method before discussing the results.

Method

We selected recordings of four participants: one video recording and one Kinect recording per participant. To test the impact of quality and length of recordings on analyses, we selected two short and two long recordings with high and low tracking ratios each. Table 1 shows a comparison.

To investigate *multimodal synonyms* [12], i.e. using gestures or speech commands to invoke the same browser function, we performed three analysis tasks on the recordings: **1) Gesture analysis task:** Find all occurrences of flick right gestures. **2) Speech analysis task:** Find all occurrences of “back” commands. **3) Gesture/speech analysis task:** Determine gesture/speech ratio of backward and forward interactions such as flick hands or “go back/forward” for navigating the page history or “previous/next tab” for switching tabs.

The comparison was conducted in three steps. First, as ground truth, we conducted macro-level video analysis using the camera recordings from Task 3. Second, we created gesture and audio scripts using KinectScript and used them for the three analysis tasks. Third, we aggregated the obtained results and compared the two methods using the number of detected events as the main metric. The time taken for each analysis task was recorded and used as a secondary metric.

ID	Method	Gesture	Speech	Gesture/Speech	Total
P8	Manual	0	9	11	20
	KinectScript	0	5	7	12
P11	Manual	0	4	7	11
	KinectScript	0	3	4	7
P16	Manual	8	1	10	19
	KinectScript	1	6	7	14
P22	Manual	2	0	15	17
	KinectScript	1	1	6	8

Table 2. Detected events for manual vs. KinectScript-based analysis

As commonly suggested, video analysis was performed using a media player and a spreadsheet application [2], here VLC and Excel. For Step 1, we only noted flick right gestures, for Step 2, only back speech events and, for Step 3, both gesture and speech for all forward and backward interactions. Each event was recorded together with the timestamp. If required, the video was paused during analysis, e.g. when many events occurred close to each other. After the video analysis, recordings of Kinect’s skeleton and audio streams were analysed using the scripts formulated in KinectScript. For the gesture analysis task, a custom gesture script `skeleton1 gesture flickright` was used to detect flick right gestures. As already mentioned, Kinect Analysis was extended to use the same algorithm for detecting flick right gestures as Kinect Browser. The script `skeleton1 says back` was used for the speech analysis task to detect when participants said “back” to interact with Kinect Browser. Finally, for the gesture/speech analysis task, we used two different scripts. In order to find all speech-based backward and forward interactions, we used the script `skeleton1 says back,forward,next tab,previous tab`. In addition, we used a second script `skeleton1 gesture flickright` or `skeleton1 gesture flickleft` to detect all forward and backward gesture interactions. The number of detected events as generated by the script output was noted.

Results

Table 2 summarises the results obtained for manual and KinectScript-based analysis. KinectScript only found a fraction of the events—the misses of $8/20 = 40\%$, $4/11 = 36\%$, $5/19 = 26\%$ and $9/17 = 53\%$ can be explained as follows.

The gesture scripts were not able to recognise all flick gestures detected with video analysis. This was mainly for two reasons. First, participants often demonstrated gestures repeatedly as Kinect Browser could not detect them on the first attempt. Since the gesture recognition code was shared with Kinect Analysis, also KinectScript only caught the gestures that were finally recognised by Kinect Browser. Second, accidental triggering of flick gestures, e.g. `flickleft` recognised immediately after `flickright` when returning to normal pose, produced some false positives. Also Kinect tracking quality seemed to have an impact on the quality of results. In particular, the gesture analysis task produced poor results for P16 and P22, which were the recordings with low tracking quality.

As for speech, some false positives were detected due to talking and background noise. The speech recognition engine was configured for the English language and talking sometimes occurred in languages other than English, which increased the false matches. In addition, some speech commands could not be recognised because the pronunciation was not clear or the participant’s voice was too soft.

The largest differences were in the speech analysis tasks with time savings averaging around 30% using audio scripts. Audio script output using speech recognition was almost immediately available independent of recording length, while manual analysis required the full video to be watched in realtime.

DISCUSSION

We have shown that Kinect Analysis can be useful and efficient in recording and analysing Kinect data. The fact that KinectScript produced false positives and missed events detected by manual analysis does not mean that it has no added value compared to conventional tools. In the Kinect Browser study, it proved to be a very useful tool for initial indexing. The automated coding of the timeline can save time and bootstrap the process. It can guide coders in the sense that they can easily identify the gaps between the timeline annotations after using KinectScript and streamline the manual analysis using video. For the Kinect Browser data, doing a first rough sequencing of the timelines by the referents investigated in the guessability study, and then refining the coding as well as performing additional analysis tasks using novel features such as traces and heatmaps, turned out to be quite effective. While we have not carried out detailed comparisons, the improvements to Kinect with the v2 sensor and SDK 2.0 give reason to believe that KinectScript could produce better results than the ones included in the paper.

Kinect Analysis as a method may facilitate new ways of conducting analyses. But there are two limitations to our proof of concept due to the Kinect sensor and KinectScript.

The first is that our implementation is subject to the limitations of the Kinect sensor and relies on Kinect's default skeletal tracking, which means rather basic gesture detection, no tracking of fingers, eye gaze, etc. However, the tracking capabilities and recognition accuracy of Kinect Analysis could be improved in the future based on its extension mechanisms. First, with relatively minor changes to the implementation, the new Kinect sensor with better tracking capabilities could be used. Second, more powerful motion tracking hardware, but in principle similar to Kinect, such as Leap Motion or Vicon, could be combined with Kinect Analysis. Finally, custom recognisers that make advanced use of Kinect's depth camera for better gesture recognition could be integrated.

The second limitation relates to completeness and expressiveness of the KinectScript language. While our proof of concept explored different types of scripts, future extensions should address the temporal dimension in more detail. One example that we already mentioned would be to allow scripts to specify a certain timespan for detecting the occurrence of one event close to another event, e.g. by adding a **within 2s** keyword for specifying a 2-second threshold. The Kinect Browser study also raised that it would be practical to use any coded custom event when formulating combined scripts, e.g. **#goback** and **skeleton1 gesture flickleft** for annotating flick hand gestures only for the Go Back referent after first coding at a per-referent level, which we could easily add.

In general, we see Kinect Analysis as a first example of a new class of tools that make specific use of Kinect to facilitate

user studies. Apart from organising capture and bootstrapping analysis, it could also be valuable as a tool for training and improving recognisers. We see two possible usage scenarios. First, similar to DeJaVu [9], pilot studies could be done with Kinect Analysis before a Kinect-based system is deployed. Second, as with the Kinect Browser study [16], a guessability study could be conducted for collecting interaction proposals first before developing required recognisers.

We also believe that Kinect Analysis has the potential to cater to other kinds of studies than the one in the focus of this paper. We see the possibility to conduct user studies in public spaces with Kinect sensors, rather than video cameras, being used for anonymous recording. Moreover, by extending Kinect Analysis' techniques for recording, visualising and analysing data to other devices, we see great potential for cross-device studies. To capture all interactions *on, around and between* devices, user interaction data recorded on smartphones and tablets similar to W3Touch [17] could be correlated with data recorded using Kinect Analysis. This is a promising idea we have started to explore as part of the XDKinect project [18].

CONCLUSION

In this paper, we have presented Kinect Analysis—a new system for user interaction analysis based on Kinect. Rather than just using video or custom experiment software individually developed for each study, the idea is that increasingly richer, depth camera-based motion and speech input sensing devices such as Kinect are used both for capture and analysis. In particular, Kinect's depth and skeleton streams can be visualised, annotated and queried based on new tools specifically designed to assist in the analysis.

The proof of concept presented in this paper implements tools at three levels: **a) annotation**—log file parsers and scripts that populate the timeline; **b) visualisation**—selective joints, heatmaps and traces; and **c) scripting**—skeleton, distance, gesture and audio scripts based on KinectScript. Each of these were designed with extensibility in mind. Using Kinect Analysis in a recent Kinect-based guessability study [16] demonstrated several useful features and the potential to facilitate important analysis tasks. More detailed studies with qualitative video researchers on the usability and usefulness of Kinect Analysis are planned in the future.

In the paper, we have argued that we understand Kinect Analysis as both a tool and a method. As a tool, it enables researchers to collect data on popular gesture, speech and multimodal interactions [16]. As a method, it adds to the growing body of knowledge surrounding suitable designs of guessability studies [12, 28]. We plan on making Kinect Analysis available under an open-source license and also setting up a web site for the HCI community to try it out and share their data and experience. This could generally be useful to those developing Kinect-based systems and conducting guessability studies. It could also provide a valuable basis for developing design patterns for natural user interfaces, something that our community is still relatively short of. Finally, we believe that this could be an important step forward in promoting replication and making it feasible.

CODE AND DATA

The Kinect Browser study material including the proposed multimodal interaction sets and source code are available from <https://github.com/globis-ethz/kinectbrowser>. The Kinect Analysis source code and some of the anonymous recordings used in this paper are available from <https://github.com/globis-ethz/kinectanalysis>.

REFERENCES

1. Ashbrook, D., and Starner, T. MAGIC: A Motion Gesture Design Tool. In *Proc. CHI* (2010).
2. Burr, B. VACA: a tool for qualitative video analysis. In *Proc. CHI EA* (2006).
3. Fouse, A., Weibel, N., Hutchins, E., and Hollan, J. D. ChronoViz: A System for Supporting Navigation of Time-coded Data. In *Proc. CHI EA* (2011).
4. Heath, C., Hindmarsh, J., and Luff, P. *Video in Qualitative Research*. SAGE, 2010.
5. Hinrichs, U., and Carpendale, S. Gestures in the wild: studying multi-touch gesture sequences on interactive tabletop exhibits. In *Proc. CHI* (2011).
6. Hoste, L., and Signer, B. SpeeG2: a speech- and gesture-based interface for efficient controller-free text input. In *Proc. ICMI* (2013).
7. Jang, S., Elmqvist, N., and Ramani, K. GestureAnalyzer: Visual Analytics for Pattern Analysis of Mid-Air Hand Gestures. In *Proc. SUI* (2014).
8. Jones, B. R., Benko, H., Ofek, E., and Wilson, A. D. Illumiroom: peripheral projected illusions for interactive experiences. In *Proc. CHI* (2013).
9. Kato, J., McDirmid, S., and Cao, X. DejaVu: Integrated Support for Developing Interactive Camera-Based Programs. In *Proc. UIST* (2012).
10. Klemmer, S. R., Hartmann, B., and Takayama, L. How Bodies Matter: Five Themes for Interaction Design. In *Proc. DIS* (2006).
11. Lee, S.-S., Chae, J., Kim, H., Lim, Y.-K., and Lee, K.-P. Towards more Natural Digital Content Manipulation via User Freehand Gestural Interaction in a Living Room. In *Proc. UbiComp* (2013).
12. Morris, M. R. Web on the Wall: Insights from a Multimodal Interaction Elicitation Study. In *Proc. ITS* (2012).
13. Morris, M. R., Danielescu, A., Drucker, S. M., Fisher, D., Lee, B., m. c. schraefel, and Wobbrock, J. O. Reducing Legacy Bias in Gesture Elicitation Studies. *Interactions* 21, 3 (2014).
14. Morris, M. R., Wobbrock, J. O., and Wilson, A. D. Understanding Users Preferences for Surface Gestures. In *Proc. GI* (2010).
15. Nacenta, M. A., Kamber, Y., Qiang, Y., and Kristensson, P. O. Memorability of Pre-designed and User-defined Gesture Sets. In *Proc. CHI* (2013).
16. Nebeling, M., Huber, A., Ott, D., and Norrie, M. C. Web on the Wall Reloaded: Implementation, Replication and Refinement of User-Defined Interaction Sets. In *Proc. ITS* (2014).
17. Nebeling, M., Speicher, M., and Norrie, M. C. W3Touch: Metrics-based Web Page Adaptation for Touch. In *Proc. CHI* (2013).
18. Nebeling, M., Teunissen, E., Husmann, M., and Norrie, M. C. XDKinect: Development Framework for Cross-Device Interaction using Kinect. In *Proc. EICS* (2014).
19. North, C., Dwyer, T., Lee, B., Fisher, D., Isenberg, P., Robertson, G. G., and Inkpen, K. Understanding Multi-touch Manipulation for Surface Computing. In *Proc. INTERACT* (2009).
20. Oh, U., and Findlater, L. The Challenges and Potential of End-User Gesture Customization. In *Proc. CHI* (2013).
21. Ruiz, J., Li, Y., and Lank, E. User-Defined Motion Gestures for Mobile Interaction. In *Proc. CHI* (2011).
22. Schmidt, D., Seifert, J., Rukzio, E., and Gellersen, H. A Cross-Device Interaction Style for Mobiles and Surfaces. In *Proc. DIS* (2012).
23. Seyed, T., Burns, C., Sousa, M. C., Maurer, F., and Tang, A. Eliciting Usable Gestures for Multi-Display Environments. In *Proc. ITS* (2012).
24. Troiano, G. M., Pedersen, E. W., and Hornbæk, K. User-Defined Gestures for Elastic, Deformable Displays. In *Proc. AVI* (2014).
25. Vatavu, R. User-Defined Gestures for Free-Hand TV Control. In *Proc. EuroITV* (2012).
26. Weibel, N., Emmenegger, C., Lyons, J., Dixit, R., Hill, L. L., and Hollan, J. D. Interpreter-Mediated Physician-Patient Communication: Opportunities for Multimodal Healthcare Interfaces. In *Proc. PervasiveHealth* (2013).
27. Weichel, C., Lau, M., Kim, D., Villar, N., and Gellersen, H. MixFab: A Mixed-Reality Environment for Personal Fabrication. In *Proc. CHI* (2014).
28. Wobbrock, J. O., Morris, M. R., and Wilson, A. D. User-Defined Gestures for Surface Computing. In *Proc. CHI* (2009).