bRIGHT – Workstations of the Future and Leveraging Contextual Models

Rukman Senanayake(✉), Grit Denker, and Patrick Lincoln

SRI International, 333 Ravenswood Ave, Menlo Park, CA, USA
{rukman.senanayake,grit.denker,
patrick.lincoln}@sri.com

Abstract. Experimenting with futuristic computer workstation design and specifically tailored application models can yield useful insights and result in exciting ways to increase efficiency, effectiveness, and satisfaction for computer users. Designing and building a computer workstation that can track a user's gaze; sense proximity to the touch surface; and support multi-touch, face recognition etc meant overcoming some unique technological challenges. Coupled with extensions to commonly used applications to report user interactions in a meaningful way, the workstation will allow the development of a rich contextual user model that is accurate enough to enable benefits, such as contextual filtering, task automation, contextual auto-fill, and improved understanding of team collaborations. SRI's bRIGHT workstation was designed and built to explore these research avenues and investigate how such a context model can be built, identify the key implications in designing an application model that best serves these goals, and discover other related factors. This paper conjectures future research that would support the development of a collaborative context model that could leverage similar benefits for groups of users.

Keywords: Contextual model · Cognitive model · Task automation
Multimodal input · Gaze tracking

1 Introduction

1.1 Motivation

What features and capabilities would future computer workstations have? How would they impact our productivity and ability to solve complex problems? Will they help us collaborate more effectively? Is it possible to achieve a significant leap in performance without changing the application models being used today? These are some of the questions we asked ourselves when we set out to experiment on futuristic workstation design. We were motivated by the ready availability of unused computational power (most PCs have CPUs that do not function to full capacity under normal use), rapid growth in the availability of RAM and hard drive storage, the wide availability of inexpensive sensor systems, and the clear understanding that we do not use all these resources in PCs a vast majority of the time. We were also interested in seeing if we could raise microprocessor features such as instruction pipelining and branch prediction

S. Yamamoto and H. Mori (Eds.): HIMI 2018, LNCS 10904, pp. 346–357, 2018.
https://doi.org/10.1007/978-3-319-92043-6_29

to the user level tasks, what the system requirements would be to achieve that, and whether these features would significantly increase a user's efficiency and effectiveness.

1.2 Design Rationale

From the very start of this project, we were interested in experimenting with multi-touch as a primary input model. We wanted the main input area of the workstation to be fairly large so that more than one person could work together (to support possible future experiments with multiple users simultaneously engaging with the system). We selected gaze-tracking as one of the key input models to the new workstation because of its ability to help us understand the user's context. We wanted to experiment with building a system that could put screen controls (like Ok or Cancel buttons) directly under a user's hands, shifting the traditional paradigm of reaching or searching for the controls. This line of investigation revealed that proximity detection (detecting the nearness of the user's hands to the main input surface) would be a valuable capability.

We also determined that capturing user-level semantics during interactions with the system would significantly improve the usefulness of information related to gaze tracking as well as interactions on the touch screen. Therefore, we extended the applications used for testing so that they can report accurately about on-screen items of interest with which the user engages. For areas of research such as task automation and contextual/cognitive auto-fill, we introduced a semantically enhanced interaction model for the applications being used to enable the capture of a user's workflows while preserving user-level operational semantics. For example, if the user pressed the Send button in a mail client, our system would respond, "The user executed the send mail function in Thunderbird with the following parameters…" instead of reporting a less meaningful output such as "Send button pressed".

To facilitate research about team dynamics and methods for improving collaboration, we decided to build a centralized server modeled on massively multi-player online role playing (MMORPG) game systems. This approach would enable us to capture the context of all users irrespective of their location in real time, reason about it, and then adjust the user's experience based on the analysis results.

2 Related Work

The development of bRIGHT involved R&D in various areas of HCI. In this section we will be looking at some of those areas such as multimodal input, gaze tracking and context modeling. bRIGHT utilizes semantic interaction and visualization models to capture the end user's context in a very rich fashion, see [1] for a detailed discussion on how this is achieved.

Gaze Tracking (GT) is the process of detecting gaze coordinates on a display or the point where a person is looking at. For bRIGHT it was fundamentally critical that we support gaze tracking in a fairly unusual configuration. Since the bRIGHT workstation forces an end user to stand about four feet away from the vertically mounted primary screen they are looking at, we found it important to design a system with various

features that are not in typical GT systems. For example, since the user is standing in front of a large screen, we felt it was critical that our GT system allows the user to turn their head naturally when looking at the edge of the screen. This also meant the 'head box' (the area within which a user's gaze is effectively tracked) is much larger in our system than conventional approaches. Gaze tracking recently has been making important contributions in research in conventional HCI [3–5] as well as human cognitive studies [6–11].

Many explorations have been made into developing both nonintrusive [12–20] and intrusive GT techniques [21–23] in order to develop more accurate, efficient and user friendly systems. Nonintrusive methods have been much preferred over its intrusive counterpart due to the potential comfort when performing EGT. Compared to intrusive techniques, image/video processing lies at the core of nonintrusive methods where users are not prodded with electromechanical hardware (e.g. head-mounted cameras, lenses, etc.).

Generally, the process of GT involves four main components. They are (1) head pose detection, (2) eye location detection, (3) gaze estimation (GE), and (4) area of interest analysis. In the case of bRIGHT steps (1–3) were implemented very similar to conventional approaches that are based on active illumination using Infrared. Step (4) on the other hand was radically different to most common approaches due to our ability to access the semantic visualization model of the application being used by the end user (discussed later).

In bRIGHT we grounded our approach based on the work in [24] as one of our team members was Zhiwei Zhu, who designed and built the first generation of GT for this workstation. These were then further improved upon by increasing the accuracy and updating the IR cameras to much higher fidelity.

bRIGHT is an extensible multimodal input system. In essence we have a workstation design where new input modalities can be added as needed by end user context and the systems contextual modeling and other features can dynamically adapt to these. The reason for this approach is grounded in benefits of multimodal input. Various research efforts in the past have shown that multimodal systems are preferred by users over their unimodal alternatives, resulting in higher flexibility and reliability while better meeting the needs of a variety of users [25, 26]. Such systems provide several advantages, including improved efficiency, alternating interaction techniques, and accommodation of individual differences, such as permanent or temporary handicaps [27, 28].

Industrial robot programming using markerless gesture recognition and augmented reality is evaluated in [29]. It allows the user to draw poses and trajectories into the workspace, we believe our semantic interaction models will give us the ability to extend such modalities into various domains where seamless gesture recognition would be of great value. The authors state that such systems show a significant reduction of required teach-in time. A combination of human intention and language recognition for task-based dual-arm robot programming is evaluated in [30]. They conclude that using multimodal input reduces the programming complexity and allows non-experts to move a robot easily. User group differences in modality preferences for interaction with a smart-home system are presented in [31]. Their study shows that female participants prefer touch and voice over gestures. Male subjects prefer gesture over voice. Hinckley

and Oviatt evaluate different input modalities based on various aspects like input speed or user feedback possibilities [32, 33]. By using multiple modalities, the system can make up for the shortcomings of other modalities.

3 Methodology

This section describes the various processes we used to develop both the hardware and software components of the bRIGHT workstation and framework.

3.1 Hardware Components and Sensor Platform

Multi-touch and Proximity Detection

We built all the multi-touch surfaces used in bRIGHT based on the technique pioneered by Han [2]. We selected Frustrated Total Internal Reflection (FTIR) as the multi-touch basis because our experiments showed that this method of vision-based multi-touch can be extended to support proximity detection. Being able to approximately detect where the user's hands are along the surface normal of the multi-touch plane is a key requirement of bRIGHT's ability to position controls under a user's hands. Using fairly cheap infrared emitters (near Infrared in the 825-nm to 920-nm range) and widely available machine vision cameras, allowed us to test and confirm the practical viability of this approach.

Figure 2 shows a screenshot of the raw signal processing software developed for this task. As can be seen in the image we stitch the 4 Infrared (IR) camera's inputs into a single video feed, and then apply background removal. After that OpenCV based blob detection is used to detect both touch and proximity. We use two different blob detectors for these tasks. Proximity detection requires that we configure a blob detector to be highly sensitive to IR data with a high value for maximum blob size.

Our approach significantly differs from the technique described in [2] in one significant way. When it comes to how the Infrared emitters are attached to the side of the acrylic surface which forms the main multi-touch area, we made sure that the angle of contact was such that Infrared would not only flood the acrylic but also the area above it. This creates the effect of having an Infrared light field above the multi-touch surface which in turn is useful for proximity detection using the same Infrared sensors that detect touch on the surface. Then we created a second instance of the same blob detection algorithm in OpenCV and configured the parameters to detect the user's hands as they hover close to the multi-touch surface. When configuring the blob detection algorithm we had to setup the parameters to track two fairly large blobs that had very low threshold of Infrared being reflected. This was achieved by increasing the values for amplification, lowering the values for IR threshold, increasing the values for highpass blur and increasing the values for the smallest blob detected to a size similar to a palm.

Our multi-touch software then converts detected blobs into TUIO (Tactile User Input/Output) output. We then developed a windows driver that registers our device as a USB Human Interface Device (HID) with MS Windows operating systems. This allows us to pass on the TUIO output to windows operating systems so they can be

dealt with the same way as key strokes or mouse pointer movements. This approach was selected because we wanted the multi touch aspect of our development to be useful for any application within MS Windows. And the TUIO processing is then the same as any other Windows base user interaction handling.

Gaze Tracking
The gaze-tracking system we developed has characteristics that differentiate it from most commercial gaze trackers. We were interested in developing a gaze tracker that support a user's significant head movements; i.e., the head box for the gaze tracker had to be significantly larger in volume than usual. The gaze tracker also had to track the user's gaze on a large TV instead of on a regular PC monitor. In our case, we used a 60" LCD TV (150 cm measured diagonally) as the primary display. This meant that the user was significantly further away (about 115 cm) from the surface on which the system would track gaze movements. (Typical gaze trackers work on PC monitors, which are about 75 cm measured diagonally, and the user is a maximum of about 70 cm away. Our technique is based on active infrared illumination and built on the approach discussed in [2].

Face Recognition
The active illuminated infrared system for gaze tracking also detects and recognizes the user's face. We used the OpenCV library and its support for face detection and recognition for this feature. Face recognition is the primary authentication method in bRIGHT and is how the MMORPG backend server distinguishes individual users.

Speech/Speaker Recognition
We initially incorporated speech recognition support for bRIGHT using SRI's STAR Lab's DynaSpeak technology. In many of the application domains where bRIGHT will be useful (such as Battle Management Command and Control, Intelligence analysis etc), we believe voice commands and speech input will have a significant role to play. But at present our focus is on interaction models associated with the multi touch surface and proximity detection. Therefore speech and speaker recognition features are not being currently utilized or experimented with, but may be used in the future if a need arises.

3.2 Software Components

Application Models and Extensions
Maximizing the benefits of integrated gaze tracking in this workstation required a technique that would allow us to identify and label (mark up) all the on-screen constructs of an application that may be of interest to a user. To satisfy this requirement, we developed a Semantic Visualization Model (SVM) for an application. The SVM is a collection of markup that captures user-level semantics of on-screen constructs. This is done only for controls or screen components that are of interest and are used frequently. For example, we do not markup controls such as a scroll bar in a user interface.

To experiment with application modeling approaches and ways to capture user's workflows at a highly detailed level, we needed an approach that would record interactions without losing user-level operational semantics. To support this requirement, we introduced the concept of a Semantic Interaction Model (SIM) for an application.

The SIM is a collection of markup that describes the operational semantics of possible user interactions while using the application. Refer to [1] for a more in depth discussion of this topic and examples of how this is used in a cyber-security context.

MMORPG-Based Design

For selecting support for multiple users, we based our application model and backend server design on techniques used in MMORPGs such as World of Warcraft. The MMORPG model will allow us to experiment in the future with team dynamics and investigate ways to effectively study collaboration in geographically disparate groups. We collaborated with a commercial game server developer Electrotank (www.electrotank.com) and studied the design of their ElectroServer product, which is used in commercial MMORPG style games. We used this design as the base for the design of bRIGHT's backend server.

The backend server handles every single user's input to bRIGHT systems (shown in Fig. 1[b]). The input gathered from users are instances of the SVM and SIM that pertain to the applications being used by the end user. The server then forwards these updates to the Contextual Model (CM), which attempts to build a model of the user's current context based on these inputs.

Fig. 1. bRIGHT workstation and its main components: (a) multi-touch surface, (b) backend server, (C) gaze tracker, (D) Contextual Model

For every single user, that utilizes a bRIGHT system, the backend server creates an individual instance of that user in the bRIGHT universe, and maintains 2 types of information: (a) pre-cognitive information (information that the bRIGHT workstation has already been exposed to but the user has not, (b) SIM and SVM instances generated by the user engaging with the bRIGHT system. An example of pre-cognitive information might be an email the user has received, but has not yet read, but the bRIGHT system has already read and processed this (since it has access to all the data feeds to the user). In such cases, the information is the pre-cognitive information is evaluated to

see if it is relevant to the current user's context. If it is deemed so, then the bRIGHT system can take steps to alert the user to such information.

Contextual Model

The Contextual Model (CM) in bRIGHT, for now, is built per individual user, based on her gaze (SVM instances) and interaction (SIM instances) with the system. The CM applies rules and weights to data elements to which the user has been exposed. Figure 2 shows a sample CM after a user has been performing a few tasks in two cyber security related applications.

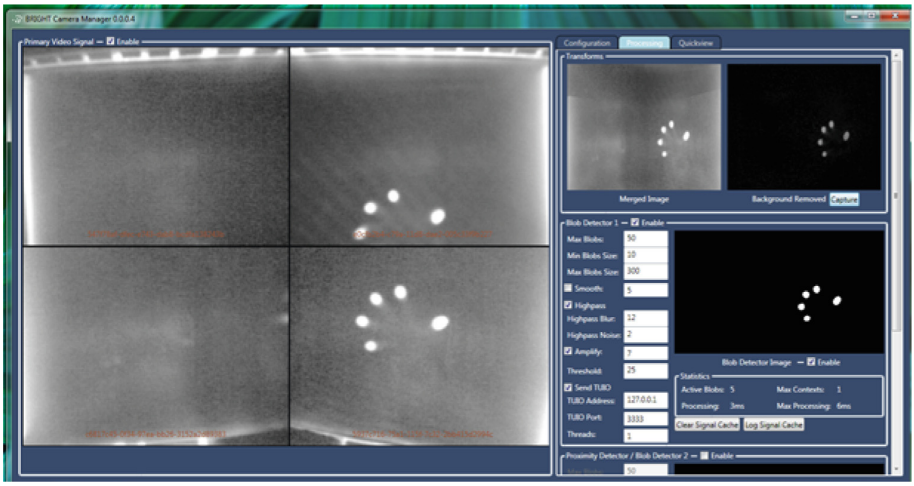

Fig. 2. FTIR based multi-touch processing software developed using OpenCV

The CM can grow extremely large, spanning hundreds to thousands of components due to the volume of information a typical user engages in. The size of an individual node represents how the rules and weights in the CM evaluate the significance of a particular item at this time in the user's context. In Fig. 2 the IP address

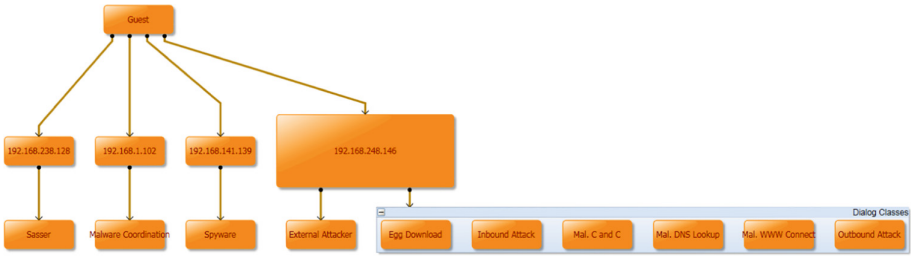

Fig. 3. Sample Contextual Model after a few actions from the user

192.168.248.146 has much more significance in the current context than other items in the CM. Refer to [3] for a more in-depth look at how the CM is built and its impact (Fig. 3).

The contextual model is built primarily based on what we refer to as "contextual indexes". Contextual indexes are references that tie various data elements in a user's context together based on relationships reported due to application characteristics or known properties in the ontological definition of the SIM or SVM instance associated with the data. As can be imagined as a user goes through a multitude of applications and vast amount of information during the course of a regular work day, the contextual index set generated by them tends to be extremely large. This causes the contextual model to be a highly complex map (connected component) that has a vast array of data and attributes being bound together in various relationships reflecting the end user's context.

4 Discussion

4.1 Current Focus of Research

Figure 4 lays out our envisioned technology development road map for bRIGHT. We are currently experimenting with the CM and working on contextual auto-fill.

We are no longer pursuing research in Learning by Demonstration or in developing a bRIGHT desktop. The learning by demonstration approach yielded results that were not flexible enough to support highly dynamic use cases such as cyber security operations, where the same task may be performed in several similar but different ways. We abandoned the experiments on building a desktop that could leverage bRIGHT's features when we realized that the CM should ideally be used as a basis for any future desktop. The lessons learned point us in the direction of a contextual desktop where the user's current context or mental model is the primary interface of the system. Therefore, we have halted the work on incremental desktop design until the CM and its implications and potential are fully realized.

Since the CM is built to reflect the user's context and figure out the significant items of interest within that context, the CM can be used to build a highly accurate auto-fill feature for most applications. This is similar to the auto filling most users experience when they use a web search engine such as Google. In the case of bRIGHT, however, auto fill can be done for any data field in any bRIGHT-enabled application. In Fig. 4 the contextual auto-fill popup (A) appears when the user selects the Get IP Reputation field in the Infected World application, since this field requires the user to input an IP address. Given the user's current context, the CM informs the bRIGHT framework when a user needs an item of type IP address to use this collection of IP addresses and also sends additional information, such as where the user has interacted with these IP address before (From column). If the user has not yet seen this item, it will be marked as pre- cognitive (as in the group with the green check mark). It is possible for bRIGHT to include items that the user has not yet seen; for example, the user receives an email that is not yet open but has been processed by bRIGHT. In such a circumstance, if there is a relevant item that bRIGHT can link to current context (by

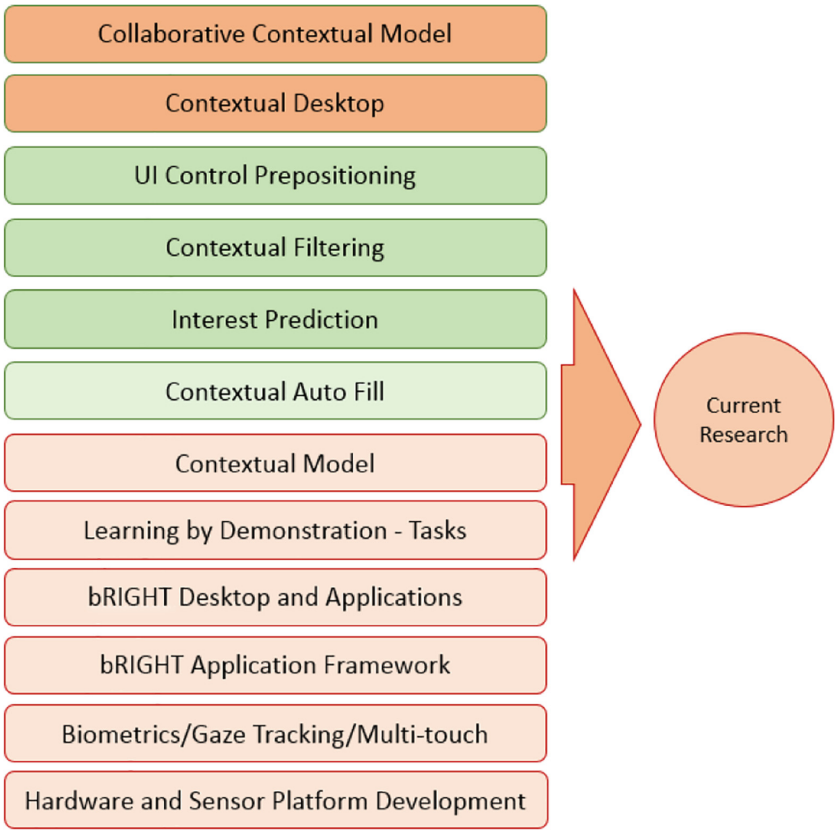

Fig. 4. bRIGHT technology research road map

type or other contextual link), bRIGHT will extract that and place it in the contextual auto-fill popup and mark it as pre-cognitive.

We are also investigating how to improve the semantic markup process. Writing plug-ins to applications or extending their code base to support bRIGHT is a significant hurdle for wider adoption of our approach. Therefore we are at present investigating how to automate this process significantly by developing tools that would allow us to markup an application by linking the semantic interaction model and semantic visualization model constructs to the applications on screen controls and widgets. This reduces the overhead associated with deploying applications on bRIGHT systems and getting them to interact correctly with the backend server and other signal generating components. It also reduces the engineering development overhead associated with adding new tools or software to the platform.

4.2 Conclusion

bRIGHT as an experimental HCI platform that has been an exciting development so far. The features we are experimenting with at present, such as contextual auto-fill and the ability to successfully implement them encourages us to pursue more ambitious research goals such as contextual filtering. Contextual filtering is a great way to manage an end user's cognitive load when they are performing critical tasks. But the risk of such an approach is that if we filter out data that the user needs to solve a problem in the current context, then we will add to the challenges they are facing rather than alleviate them. Whether that can be successfully achieved depends to a great degree on the accuracy of the contextual model we can build and the inferences we can derive from it. At present, in terms of interactions with the system, bRIGHT does not lose any vital information when recording the user's context. There is a degree of uncertainty introduced into our modeling approach due to gaze tracking and how we interpret that in the user's context but we are confident that we can compensate for that and still achieve the level of accuracy we strive for such that the contextual model can be used to increase the user's effectiveness, efficiency and reduce human errors.

References

1. Senanayake, R., Denker, G.: Towards more effective cyber operator interfaces through semantic modeling of user context. In: Nicholson, D. (ed.) Advances in Human Factors in Cybersecurity, vol. 501, pp. 19–31. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41932-9_3
2. Han, J.Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology. ACM (2005)
3. Bulling, A., Gellersen, H.: Toward mobile eye-based human computer interaction. IEEE Pervasive Comput. **9**(4), 8–12 (2010)
4. Rantanen, V., Vanhala, T., Tuisku, O., Niemenlehto, P.-H., Verho, J., Surakka, V., Juhola, M., Lekkala, J.: A wearable, wireless gaze tracker with integrated selection command source for human-computer interaction. IEEE Trans. Inf Technol. Biomed. **15**(5), 795–801 (2011)
5. Corcoran, P.M., Nanu, F., Petrescu, S., Bigioi, P.: Real-time eye gaze tracking for gaming design and consumer electronics systems, pp. 347–355 (2012)
6. Bolmont, M., Cacioppo, J.T., Cacioppo, S.: Love is in the gaze: an eye-tracking study of love and sexual desire. Psychol. Sci. **25**, 1748–1756 (2014)
7. Judd, T., Ehinger, K., Torralba, A.: Learning to predict where humans look. In: ICCV, pp. 2106–2113 (2009)
8. Senju, A., Johnson, M.H.: The eye contact effect: mechanisms and development. Trends Cogn. Sci. **13**(3), 127–134 (2009)
9. Tylén, K., Allen, M., Hunter, B.K., Roepstorff, A.: Interaction vs. observation: distinctive modes of social cognition in human brain and behavior? A combined fMRI and eye-tracking study. Front. Hum. Neurosci. **6**, 331 (2012)
10. Kochukhova, O., Gredeba, G.: Preverbal infants anticipate that food will be brought to the mouth: an eye tracking study of manual feeding and flying spoons. Child Dev. **81**(6), 1729–1738 (2010)

11. Kano, F., Tomonaga, M.: How chimpanzees look at pictures: a comparative eye-tracking study. Proc. Biol. Sci. **276**(1664), 1949–1955 (2009)
12. Bergasa, L.M., Nuevo, J., Sotelo, M.A., Barea, R., Lopez, M.E.: Real-time system for monitoring driver vigilance. IEEE Trans. Intell. Transp. Syst. **7**(1), 63–77 (2006)
13. Qi, Y., Wang, Z., Huang, Y.: A non-contact eye-gaze tracking system for human computer interaction. In: 2007 International Conference on Wavelet Analysis Pattern Recognition, pp. 68–72, November 2007
14. Hennessey, C., Lawrence, P.: Noncontact binocular eye-gaze tracking for point-of-gaze estimation in three dimensions. IEEE Trans. Biomed. Eng. **56**(3), 790–799 (2009)
15. Iqbal, N., Lee, H., Lee, S.-Y.: Smart user interface for mobile consumer devices using model-based eye-gaze estimation. IEEE Trans. Consum. Electron. **59**(1), 161–166 (2013)
16. Guestrin, E.D., Eizenman, M.: General theory of remote gaze estimation using the pupil center and corneal reflections. IEEE Trans. Biomed. Eng. **53**(6), 1124–1133 (2006)
17. Hennessey, C., Noureddin, B., Lawrence, P.: Fixation precision in high-speed noncontact eye-gaze tracking. IEEE Trans. Syst. Man. Cybern. Part B (Cybern.) **38**(2), 289–298 (2008)
18. Nawaz, T., Mian, M., Habib, H.: Infotainment devices control by eye gaze and gesture recognition fusion. IEEE Trans. Consum. Electron. **54**(2), 277–282 (2008)
19. Asteriadis, S., Karpouzis, K., Kollias, S.: Visual focus of attention in non-calibrated environments using gaze estimation. Int. J. Comput. Vis. **107**(3), 293–316 (2013)
20. Zhu, Z., Ji, Q.: Novel eye gaze tracking techniques under natural head movement. IEEE Trans. Biomed. Eng. **54**(12), 2246–2260 (2007)
21. Xia, D., Ruan, Z.: IR image based eye gaze estimation. In: Eighth ACIS International Conference Software Engineering Artificial Intelligence Networking, Parallel/Distributed Computing (SNPD 2007), vol. 1, pp. 220–224, July 2007
22. Nguyen, Q.X., Jo, S.: Electric wheelchair control using head pose free eye-gaze tracker. Electron. Lett. **48**(13), 750 (2012)
23. Rae, J.P., Steptoe, W., Roberts, D.J.: Some implications of eye gaze behavior and perception for the design of immersive telecommunication systems. In: 2011 IEEE/ACM 15th International Symposium on Distributed Simulation Real Time Applications, pp. 108–114, September 2011
24. Panwar, P., Sarcar, S., Samanta, D.: EyeBoard: a fast and accurate eye gaze-based text entry system. In: 2012 4th International Conference on Intelligent Human Computer Interaction, pp. 1–8, December 2012
25. Ji, Q., Zhu, Z.: Eye and gaze tracking for interactive graphic display. In: Proceedings of the 2nd International Symposium on Smart Graphics. ACM (2002)
26. Dahlback, N., Jönsson, A., Ahrenberg, L.: Wizard of Oz studies - why and how. Knowl.-Based Syst. **6**(4), 258–266 (1993)
27. Cohen, P.R., Johnston, M., McGee, D., Oviatt, S.: The efficiency of multimodal interaction: a case study. In: International Conference on Spoken Language Processing (ICSLP), Sydney, Australia (1998)
28. Oviatt, S., Lunsford, R., Coulston, R.: Individual differences in multimodal integration patterns: what are they and why do they exist? In: Conference on Human Factors in Computing Systems (CHI), New York, USA (2005)
29. Ruiz, N., Chen, F., Oviatt, S.: Multimodal input. In: Thiran, J.-P., Marques, F., Bourlard, H. (eds.) Multimodal Signal Processing, pp. 231–255 (2010). Chapter 12
30. Turk, M.: Multimodal interaction: a review. Pattern Recogn. Lett. **36**, 189–195 (2014)
31. Perzylo, A., Somani, N., Profanter, S., Rickert, M., Knoll, A.: Toward efficient robot teach-in and semantic process descriptions for small lot sizes. In: Robotics: Science and Systems (RSS), Workshop on Combining AI Reasoning and Cognitive Science with Robotics, Rome, Italy (2015)

32. Akan, B., Ameri, A., Cürüklü, B., Asplund, L.: Intuitive industrial robot programming through incremental multimodal language and augmented reality. In: International Conference on Robotics and Automation (ICRA), Shanghai, China (2011)
33. Stenmark, M., Nugues, P.: Natural language programming of industrial robots. In: International Symposium on Robotics (ISR), Seoul, Korea (2013)
34. Stenmark, M., Malec, J.: Describing constraint-based assembly tasks in unstructured natural language. In: World Congress of the International Federation of Automatic Control (IFAC) (2014)