Article:
Jowers, I, Prats, M, McKay, A et al. (1 more author) (2013) Evaluating an eye tracking interface for a two-dimensional sketch editor. Computer-Aided Design, 45 (5). 923 - 936. ISSN 0010-4485

https://doi.org/10.1016/j.cad.2013.01.006

Evaluating an Eye Tracking Interface for a Two-Dimensional Sketch Editor

Iestyn Jowers[1]

University of Leeds, Leeds, LS2 9JT, UK

Corresponding Author

Email: iestyn.jowers@pe.mw.tum.de

Tel: +49 89 289 15154

Fax: +49 89 289 15144

Miquel Prats

The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK

Alison McKay

University of Leeds, Leeds, LS2 9JT, UK

Steve Garner

The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK

---

[1] Present address: Technische Universität München
Institute of Product Development
Boltzmannstraße 15, 85748 Garching, Germany

Evaluating an Eye Tracking Interface for a Two-Dimensional Sketch Editor

Abstract

Throughout the history of CAD, a variety of interfaces have been employed with the aim of supporting designers' construction and manipulation of digital models. This paper explores the potential of eye tracking as a CAD interface, and presents a prototype that uses collected gaze data to support shape exploration in a two-dimensional vector-based sketch editor. The eye tracking interface uses the visual interactions of users to identify interpretations of constructed shapes, according to recognised parts identified using methods from shape grammar research. It can therefore be argued that it supports construction and manipulation of shapes according to user intent. The prototype has been evaluated in a user study where design students carried out controlled shape exploration exercises which involved interpretation and manipulation of shapes. In the paper, key concepts of eye tracking methodology are introduced; the methods used to implement the eye tracking interface are described; and the user study and its outcomes are reported. The results are positive and indicate the potential for eye tracking as an interface for supporting shape exploration in CAD.

Research Highlights

- Explores the potential of eye tracking as an interface for computer aided-design
- Describes methods for inferring user intent with respect to digital shape manipulation, based on gaze data
- Presents a prototype eye tracking interface that supports dynamic shape interpretation in a vector-based sketch editor
- Reports the outcomes of a user study that evaluates how successful the eye tracking interface is at inferring users' intent with respect to shape manipulation

Keywords

eye tracking; shape interpretation; shape exploration; shape grammars

1. Introduction

Computer-aided design (CAD) systems are ubiquitous in modern design processes. But, despite this, it has been noted that they are rarely used to design, and are instead mostly used to record and communicate the outcomes of design activities [1]. Pen(cil) and paper remain the media of choice for ideation and creative design [2]. For example, Robertson and Radcliff [3] report the outcomes of a survey that questioned CAD users concerning the impact of CAD on creative problem solving. They found that use of CAD results in enhanced visualisation and communication but also an increase in circumscribed thinking and premature fixation. Indeed, Lawson and Locke [4] argue that CAD systems are not capable of supporting creative design because their development has placed too much emphasis on graphical representation techniques, and not enough emphasis on exploring how processes essential to creative design can be supported. They argue that central to creative design are ambiguity, uncertainty and parallel lines of thought and, while these are readily supported by hand drawing, they are not supported by computational systems. Similarly, Hanna and Barber [5] argue that the computerised design process supported by CAD systems does not reflect the *analysis-synthesis-evaluation-presentation* structure that is typically expected in a conventional design process. They found that although CAD presented many advantages in terms of simulation, analysis and presentation, it also resulted in a modified approach to designing. Stacey and Eckert [6] suggest that CAD systems encourage conservative design practice because they make it easier to use particular design elements, standard choices and default parameter values. A similar argument is also presented by Mitchell [7] who notes that rather than CAD systems supporting users' creative processes, users learn to adapt their design practice so that they are consistent with the particular CAD systems that they use, and this is often evident in the resulting designs. Specifically, designers have to learn to work with the data structures on which CAD systems are built, the operations and tools that the systems support, and the interactions afforded by their user interfaces.

Jenkins [8] explores difficulties that arise when using CAD systems, and suggests that these occur as a necessary consequence of the wide ranging functionality that is expected of such systems. It is proposed that difficulties could be mitigated via efficient documentation, comprehensive training and interoperability between different systems. While these are practical suggestions, they avoid addressing the intrinsic limitations of

CAD systems as tools for supporting creative design. Indeed, if future CAD systems are to realise their raison d'etre as computational aids for design, then their fundamental architecture needs to be reconsidered so that they support the processes that are essential to creative design. Accordingly, Sener and Wormald [9] report a study that sought to identify how CAD systems and their user interfaces could be re-imagined so that they can support creative design. They identified four general requirements that could lead to CAD systems that would better support designers' needs. These are: (1) bulk form creation; (2) control of form creation; (3) ease of form creation; (4) life-like form creation. This paper is concerned with the third of these requirements, "ease of form creation", and explores the potential of eye tracking as an interface to support the construction and manipulation of designed shapes. It presents a prototype interface that augments traditional mouse and keyboard input. It uses collected gaze data to support shape exploration in a two-dimensional vector-based sketch editor by considering the user's attention and intention as they computationally manipulate shapes.

Throughout the history of CAD, a variety of interfaces have been employed with the aim of supporting designers' interactions with design models, including interfaces that respond to traditional sketching, e.g. Sutherland [10], or natural actions in virtual worlds, e.g. Israel et al. [11], or interfaces that take advantage of manipulation of physical artefacts and tactile feedback e.g. Abdelmohsen and Do [12].This paper reports research that investigates the use of eye tracking as an interface for CAD, specifically for a two-dimensional sketch editor. Eye tracking technologies can be used to provide insight into visual interactions between a viewer and a scene, via recorded eye movements. Such data can identify the point of gaze of the viewer, which in turn is indicative of how the viewer is attending to a scene [13]. As an interface for CAD, eye trackers have the potential to support visual interaction between the designer and the design representation. They can act as an unobtrusive and natural interface that responds to the intentions of the designer, as exhibited by visual attention. Potentially, they can realise the ideal drawing editor tool which, in the words of Saund and Moran [14], "*would be able to read the user's mind (his visual system in particular) and synchronize with whatever image entities the user happens to perceive as significant*" (p. 175). In this way, eye tracking could substantially reduce the cognitive overhead needed for designers to interface with CAD systems and with design representations. This paper presents a prototype interface that is a first imagining of the potential applications of eye

tracking as an interface for CAD. Specifically, the prototype addresses the problem of computational shape interpretation [15], with an aim to support design generation and exploration in CAD systems.

Shape interpretation is central to the processes of design generation and exploration, and as such has been linked to creativity [16]. For example, in many creative professions, including industrial design and architecture, conceptual design involves the creation, exploration, and development of design shape alternatives. These processes are typically supported using freehand sketching because the ambiguity of sketches enhances the cognitive processes of shape interpretation which are essential in effective shape exploration and development [17]. Commercially available CAD systems offer poor support for shape interpretation because the data structures on which they are built assume that a given shape has a unique interpretation [18]. As such, they do not support the flexible interaction that designers need during design exploration for conceptual design, and instead are typically used to record the outcomes of these processes. For example, an architect exploring the shape illustrated in Figure 1a, as part of a floor plan for a building, may interpret the shape in different ways, each with its own conceptual meaning [15]. In Figure 1b, the shape is interpreted as four closed triangular wings overlooking an open quadrangle, while in Figure 1c it is interpreted as a closed square hall with four open vestibules. These are different and incompatible interpretations of the same underlying shape; incompatible because they each have a different structure according to the parts that compose them.

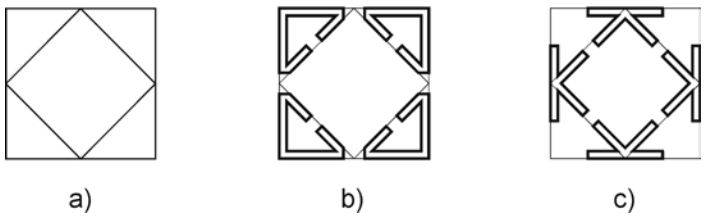

Figure 1. An ambiguous shape and two interpretations

To construct the design using CAD the architect must decide which of these interpretations to work with and, once the model has been constructed, changes in interpretation will not be readily supported. For example, if the architect constructs a model according to the interpretation in Figure 1b, then the shape will be structured according to triangular parts; but, if the model is constructed according to the interpretation in Figure 1b then it will be structured according to K-shaped parts. In

neither scenario would the CAD model support exploration of the shape because the structure of the shape will be fixed according to the parts initially identified.

The research presented in this paper explores how an eye tracking interface could support multiple interpretations of shape in CAD. It has resulted in a prototype that uses eye tracking data to identify users' interpretations of constructed shapes, based on recognised parts which are identified using methods from shape grammar research. It therefore supports manipulation of the shape according to user intention. For example, the shape in Figure 1a could be manipulated as both triangles and Ks, depending on the changing intention of the architect, as recognised by the eye tracking interface. The prototype interface recognises the differences in user intention according to the parts they are attending to, and restructures shapes so that the parts of interest are available for manipulation. The prototype has been evaluated in a user study, in which design students were tasked with carrying out controlled shape exploration exercises which involved interpretation and manipulation of shapes. The data collected was analysed to determine how successful the eye tracking interface is at inferring user intent with respect to shape manipulation. The results are positive and indicate the potential for eye tracking as an interface for CAD.

The next section presents an overview of the motivation for the research, namely supporting shape interpretation in CAD. Section 3 introduces key concepts of eye tracking methodology and a brief review of its application as an interface for computational systems. In Section 4, the eye tracking interface is introduced, with detailed descriptions of the methods used in its implementation. Section 5 reports evaluation of the interface, including a description of the user study and a discussion of the results. Finally, in Section 6, the paper ends with a brief discussion exploring the potential benefits of eye tracking as an interface for future CAD systems.

2. Supporting shape exploration with eye tracking

In this paper an eye tracking interface is presented, which was developed for a custom built two-dimensional sketch editor. The editor supports the construction and manipulation of shapes defined according to straight lines, as illustrated in Figure 2. In editors such as this, a shape is defined as a set of lines, and parts are defined according to subsets of these. Users manipulate constructed shapes by selecting and transforming

lines or the defined parts. During such manipulations, adjacent, coincident and overlapping parts often give rise to emergent and possibly unexpected forms which the user may wish to subsequently select and transform [19]. For example, in Figure 1 the shape constructed as two squares gives rise to triangular and K-shaped parts. Studies suggest that designers learn to take advantage of such emergent forms in shape exploration processes [20]. However, unless these forms are explicitly defined as parts they cannot be available for manipulation, and this necessitates restructuring the shape so that the alternative parts are defined.

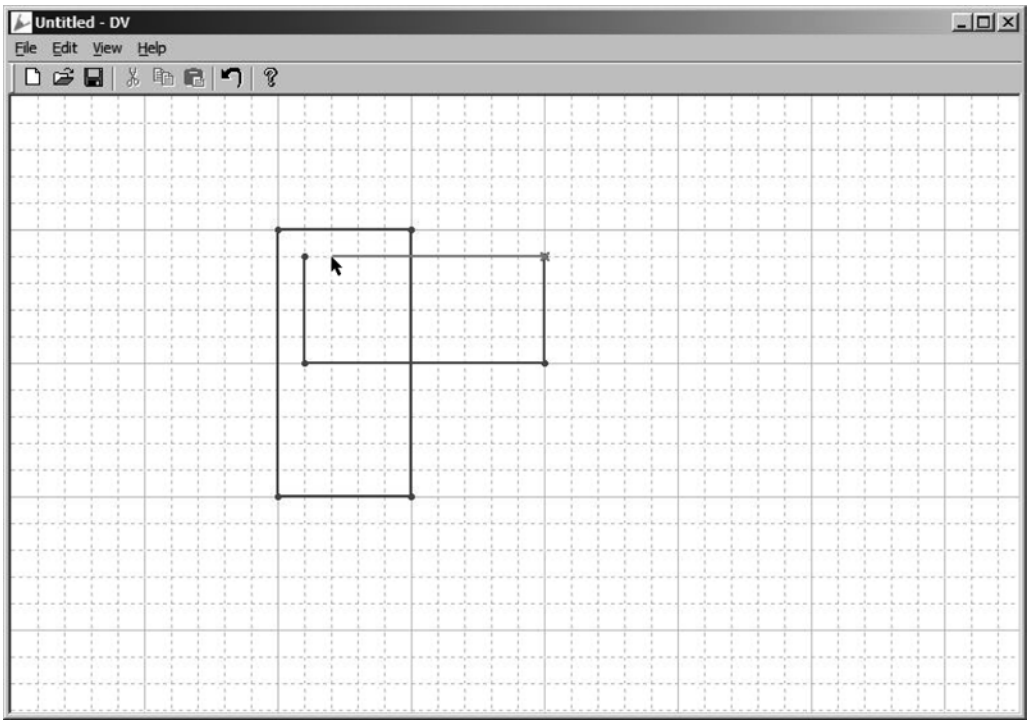

Figure 2. A custom built two-dimensional sketch editor

In commercial systems, restructuring of a shape is achieved either by redefining the parts through repeated use of operations that allow lines to be cut and/or combined and parts redefined, or by applying regularised Boolean operations of intersection, union or difference [21], or by re-constructing the shape explicitly from the required parts. These options can be laborious and time-consuming, and an alternative method that enables users to efficiently and intuitively manipulate shapes according to the parts recognised by the user at any particular moment would be beneficial for supporting computer-aided shape exploration.

Research into this problem has resulted in various methods. For example, PerSketch, is a *WYPIWYG* (What You Perceive is What You Get) drawing system that allows

perceptual interpretations of a digitally sketched shape to be specified and manipulated using simple pen-based gestures [19]. Similarly, the Back of an Envelope system is a drawing program that uses standard pattern recognition techniques to automate the recognition of emergent parts in a digital sketch [14]. Also, SD2 implements an approach based on a shape grammar formalism in which shape replacement rules are applied to identify and manipulate recognised parts of a shape [22]. The research presented in this paper describes an alternative method, in which gaze data from an eye tracker is used to infer user intention with respect to shape manipulation, and so dynamically restructure the shape according to recognised parts.

Using eye tracking to respond to users' visual interactions with shapes allows for an intuitive, dynamic system; one that supports the cognitive processes identified in shape exploration. Specifically, Schön and Wiggins [17] formalise these processes according to a *seeing-moving-seeing* model, where seeing a shape can result in its interpretation according to emergent forms or structures, which in turn informs the manipulation and construction of future shapes. A system that can respond to gaze data can infer the parts that are of interest to the user and can avoid the need for users to explicitly describe their interpretation of designed shapes. To this end, eye tracking technology presents itself as a potentially powerful interface for computer-aided design.

3. Eye tracking and visual interaction

The data collected by an eye tracker records where the foveae are directed. These are the central regions of the retinas and are responsible for gathering detailed visual information of a viewed scene. In eye tracking methods, this data is used to infer eye movements used to visually explore an object or scene. Such movements are not smooth, but are characterised by erratic jumps that occur in rapid succession and serve to direct the fovea towards different areas of visual interest [23]. The jumps, known as *saccades* (after the French for "jolt"), have durations of 10 ms to 100 ms, and are separated from each other by periods of visual *fixation*. During fixation, the foveae are directed towards a particular region of the visual stimulus for durations of 150 ms to 600 ms, allowing for detailed perception of the region. But, during fixation the eyes are not stationary; instead, fixations consist of random micro-saccades of 1 to 2 minutes of an arc in amplitude and they are inherently noisy to within an accuracy of a 1^{o} visual angle [24]. This is because movement of light is a necessary requirement for vision, and if a

visual stimulus is stabilised on the retina this results in vision fading away leaving a blank scene [23]. When combined with data concerning the position and orientation of the head, eye movement data can identify the point of gaze of a viewer as they visually explore a stimulus, as illustrated in Figure 3. Here, gaze data is represented according to nodes and arcs, where the nodes represent visual fixations, and the arcs connecting these represent saccades.

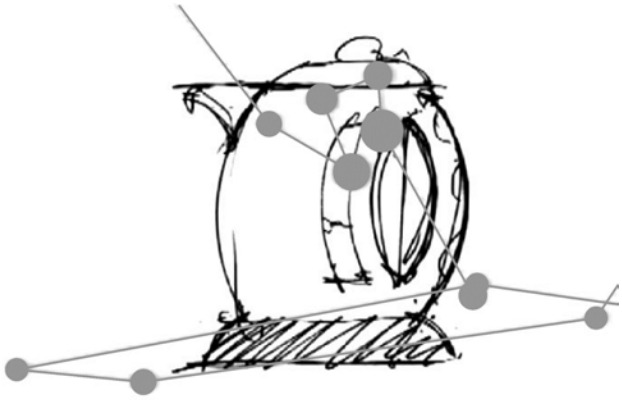

Figure 3. Example of eye tracking data

Gaze is indicative of how the viewer is attending to the stimulus [13]. In simplistic terms, attention can be considered as a perceptual filter; the mental capacity for humans to processes sensory information is limited and attention is used to focus on selections of sensory input so that the stimulus of interest can be processed. Changes in visual attention occur either sub-consciously or consciously. For example, they can occur in response to stimuli in the peripheral vision, e.g. a sudden movement; or they can occur as a result of the intention of the viewer, e.g. in preparation for a bodily movement; or they can occur as a result of active perception, e.g. when analysing an object or scene. Such changes typically result in a redirection of the gaze towards regions of interest in a visual stimulus. As such, attention is used to direct perception. However, the location of attention cannot always be measured directly [25]. In some situations, visual attention can be allocated to the peripheral vision, while conversely it is possible for the foveae to be directed towards a particular region without the viewer attending to it. Despite this, attention and gaze are closely related. For example, a voluntary saccadic movement is not possible without attendance to the region of the next fixation [26]. Also, studies have suggested that eye movements reflect and are influenced by a viewer's intent as they visually explore a visual stimulus [27]. As a result, eye tracking methodology is based on the assumption that attention and intention are evidenced through eye movements [13].

Building on this assumption, eye tracking technologies have proven to be powerful tools for investigating viewers' visual interactions with stimuli such as marketing devices [28], interfaces for computational systems [29], and the real world e.g. for controlling locomotion [30].

Eye tracking has also been explored as an alternative interface for computational systems, allowing users' visual attention to guide their interactions with on-screen objects, such as icons, menus, etc. [31]. When interacting with such objects user input is typically preceded by movement of gaze to areas of interest. As a result gaze is indicative of intended interaction and can be used as input to a human-computer interface. Simplistically, movement of an on-screen cursor can be mapped to visual saccades, while selection can be mapped to visual fixations of significant durations. Based on this, eye tracking has been applied as an interface for a range of input methods including pointing and selecting [32], text input via virtual keyboards [33] or gaze gestures [34], and drawing [35]. Such research is motivated by the requirement of providing faster, more intuitive interaction with computational systems by reducing physical movement, or by the requirement of replacing traditional input completely, for example to make interaction possible for physically impaired users, e.g. Gips et al. [36].

However, as an interface for computational systems, eye tracking suffers from three fundamental limitations [37]. Firstly, it is limited with respect to accuracy. This is because the eyes are constantly moving, resulting in noisy input data. Secondly, controlling eye movements is often not easy. This is because changes in visual attention occur in rapid succession, often sub-consciously, with the eyes fixating briefly on areas of potential interest before jumping to other areas. As a result deliberately fixating on a particular area of a screen in order to operate a user interface can take some effort. Thirdly, there is a conflict between using the eyes for input, i.e. to look at objects, and using the eyes to control, i.e. to manipulate objects. This results in unintentional actions due to the *midas touch* problem [38], so-called because the eyes are always *on*, and it is not obvious whether eye movements indicate user input, or result from the user viewing areas of the screen. Research seeks to address these limitations, with the aim to make eye tracking a fast and reliable alternative to traditional input devices. For example, limited accuracy can be been addressed by zooming the screen around the current gaze location [39], while the midas touch problem can be addressed by allowing the user to switch between different modes of visual interaction [40]. However, such solutions can

result in more complicated user interactions, requiring a certain degree of familiarisation and training.

In light of the limitations of eye tracking as a computational interface, an alternative approach to using gaze as user input has been identified: not to replace traditional methods of input, but to augment them by providing additional visual data, which is suggestive of user intent. For example, Bolt [41] explores the potential for eye tracking to support interaction within a cluttered multi-window desktop environment, by using gaze to direct organisation of the windows according to user intent. Smith and Graham [42] explore the use of eye tracking to control orientation in a video game based in a 3D environment. Similarly, Yoon and Graf [43] use an eye tracking interface to support viewing of digital 3D objects, by allowing gaze to inform the reconstruction of the objects according to where the user is looking. In each of these examples, the user looks naturally at the screen and the system takes advantage of recorded gaze to infer intended interactions with the displayed objects or environments. Trials suggest that this approach is preferable when compared to methods that require deliberate and directed visual interaction, where the users have to learn to control an interface with their eyes [44]. Accordingly, in this paper eye tracking is explored as an additional interface to augment traditional user interaction with CAD systems. The interface collects gaze data and from this infers users' interpretation of ambiguous shapes. Gaze does not have to be directed, and the interface supports fluid and natural interaction with constructed shapes, according to recognised parts.

4. An eye tracking interface

The eye tracking interface was developed in Visual C++ using the Tobii Software Development Kit[2] for the Tobii X120 eye tracker, which uses the method of *Pupil Centre Corneal Reflection* to determine eye movements [45]. This involves directing a light source of near infrared light so that it illuminates the eye and causes reflections in the pupil and the cornea, and the angle between these reflections is used to calculate gaze direction. The eye tracker is a non-intrusive, stand-alone unit that sits in front of the visual-display unit (VDU), as illustrated in Figure 4. It supports binocular tracking and a data collection rate of 120 Hz. It is accurate to within 0.5° and can compensate for drift of

[2] http://www.tobii.com/landingpads/analysis_sdk.aspx (accessed on 2/8/2012)

less than 0.3°. It includes a head-motion compensation mechanism that allows for a freedom of head movement of 0×22×30cm.

Figure 4. Illustration of the eye-tracker setup

The interface supports fluid exploration of vector-based shapes. This is achieved according to 1) consideration of users' attention at any particular moment; 2) consideration of information about potential parts of a constructed shape; and 3) consideration of users' intention with respect to selecting such parts. Based on this information, the interface is able to restructure a constructed shape according to the parts recognised by a user and make those parts available for manipulation.

4.1 Identifying attention

The eye tracking interface was developed based on a series of studies, reported in Prats et al. [46] and Jowers et al. [47]. The aim of the studies was to understand how gaze data reflects viewers' interpretations of ambiguous shapes, according to recognised parts. During the studies, participants undertook various tasks which involved viewing, interpreting and manipulating shapes, and although the number of participants was too small to provide statistically significant results, analysis of the collected data consistently identified patterns of viewing that correlated with interpretation of shapes according to specific parts.

For example, in one study participants were shown the shape in Figure 5a on two separate occasions, and on each occasion were encouraged to interpret it differently. The participants first saw the shape during a free viewing task, where they were sequentially shown a series of shapes and asked to describe them. During this task, the shape in Figure 5a was consistently described by the participants as three overlapping rectangles. The participants saw the shape a second time during a shape search task, where they were sequentially shown a series of shapes and asked to find a specific part,

the non-regular hexagon illustrated in Figure 5b. In the shape in Figure 5a, the hexagon is defined by the overlap of the three rectangles.

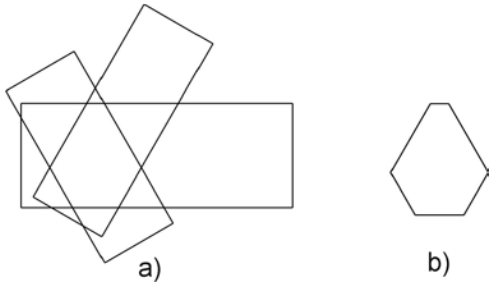

Figure 5. Example task from an eye tracking study

Comparison of the data collected for these two tasks made it possible to identify the gaze patterns that correspond with different modes of seeing: free viewing of the shape as a whole, and interpretation of the shape according to the hexagon part. The data collected for three of the participants is illustrated in Figure 6. The data in the top row was collected when participants viewed the shape as a whole and were asked to describe it. The data in the bottom row was collected when participants were encouraged to interpret the shape according to the hexagon part. The black dots illustrate visual fixations, while the grey circles illustrate the centres of gravity (COGs), i.e. the centroids, of the parts of interest to the participants: on the top, the three rectangles; on the bottom, the hexagon. Note, here the COG of a shape is calculated as the average of the vertices of the shape when represented in its maximal form.

When participants were free viewing the shape as a whole, they recognised three rectangles and the visual fixations were very loosely clustered around the COGs of these parts. But, when the participants were interpreting the shape according to the hexagon part, and paying particular attention to that part, the fixations are tightly clustered around its COG.

In general, the studies confirmed that gaze is indicative of intention, and that attention to a specific part of a shape as evidenced by visual fixation, is indicative of interpretation of the shape according to that part. It was found that attention to individual shapes, or parts of shapes, can be identified by visual fixations of long durations, exceeding 500 ms [46]. It was also found that when a shape is attended to, then visual fixations are concentrated around its COG. This confirms the findings of Vishwanath and Kowler [48], who report that participants attending to simple shapes naturally fixate on the COGs. This was also found to be true for parts of shapes; when parts were attended to, the

visual fixations were concentrated around the COGs of the attended parts [47]. These results suggest that the distance between visual fixations and COGs of parts of an ambiguous shape can be indicative of a viewer's interpretation of the shape, with respect to the recognised parts.

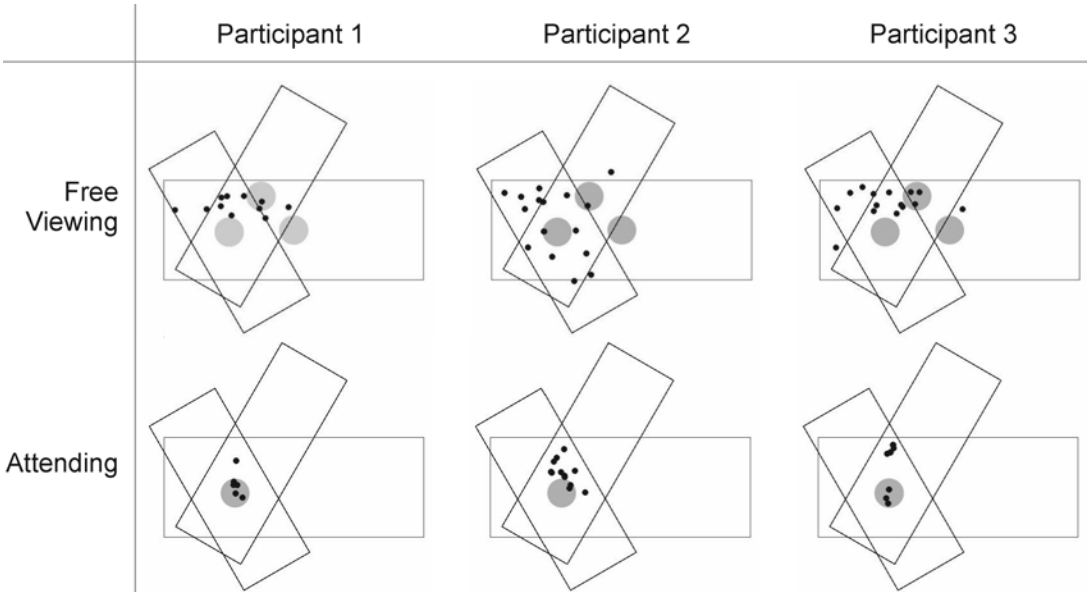

Figure 6. Example data from an eye tracking study

4.2 Potential interpretations of a shape

The eye tracking studies demonstrated that it is possible to determine a viewers' interpretation of a shape by comparing the distance between visual fixations and the COGs of its parts. Based on this, an eye tracking interface *could* infer users' interpretations of an ambiguous shape constructed in a CAD system, *if* the parts of the shape were known. A brute force approach to determining these would be to calculate all possible parts by considering the set of lines that are used to construct the shape, and restricting edges of parts to lines that connect intersection points. However, this is combinatorially explosive and computationally expensive, since a shape composed of *n* lines would contain 2^n-1 parts. For example, a shape composed of 8 lines would contain 255 parts, while a shape composed of 10 lines would contain 1023 parts. Also, many of the parts calculated using this approach may not be meaningful to a designer, for example parts composed of disconnected lines.

Instead, the approach used here is similar to that implemented by Gross [19], in which interpretation of sketched shapes is supported by matching emergent parts against shapes in a library of templates that is previously defined by the user. Here, a library of

shapes is populated by the user during shape exploration and this is used to determine the parts of a shape that may be of interest. This is illustrated in Figure 7, where a library of shapes has been added to the user interface of the two-dimensional sketch editor from Figure 2. The shape library is populated by the user on-the-fly, and can be saved for reuse in later design sessions. The library can be populated in two ways, depending on the preference of the user. Firstly, any shape that the user draws is added to the library; this allows the library to be populated quickly and intuitively with shapes that the user has explicitly constructed. Secondly, the user can use a trace function to draw over any interesting emergent parts and so add them to the library; this allows any unintended and emergent shapes that arise during an exploration process to be quickly added to the shape library. Shapes that are included in the library are considered to be of interest in the current shape exploration process, and it is assumed that any interpretation that the user makes during shape exploration will involve recognition of one of these interesting shapes as a part of the constructed shape. Accordingly, the shapes are searched for embedded within the constructed shape, resulting in a list of potentially interesting parts.

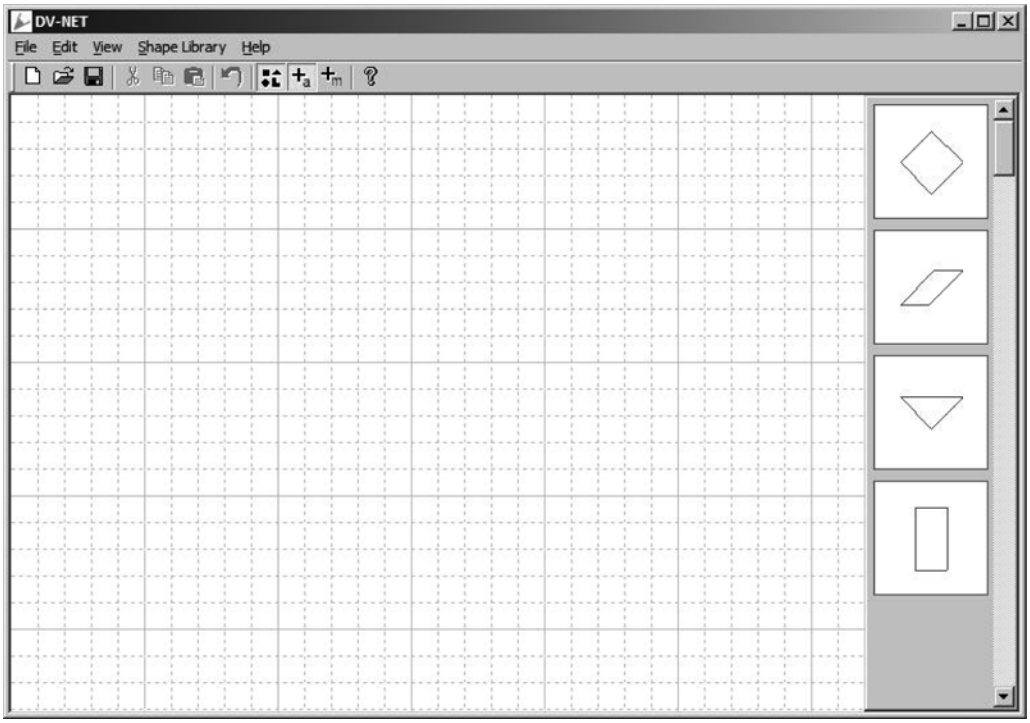

Figure 7. A sketch editor, with a library of interesting shapes

The search process implements a method of subshape detection similar to that described by Trescak et al. [49] in which rectilinear shapes are detected as parts of other

shapes under Euclidean transformations. In this method, shapes are first reduced to a unique maximal representation by merging any collinear lines that share points, i.e. lines that share end points or overlap. Maximal shapes are then compared to each other according to their intersection points, to determine transformations that embed one shape as part of the other. These include intersections of the lines explicitly defined in the shapes as well as intersections of the extended mathematical lines of which these are segments. Sets of three non-collinear points are used to deterministically calculate potential Euclidean transformations, and these are then verified by comparing the remaining points and the lines that connect them. A consequence of this method is that the shapes that can be added to the shape library must contain at least three non-collinear intersection points to ensure that Euclidean transformations can be calculated.

Comparison of the intersection points takes into consideration their spatial contexts, defined according to structures called *intersection triples* [49]. Each triple includes an intersection point, an end point of each of the intersecting lines, the angle between the intersecting lines, and the ratio of lengths of the lines. As discussed by Trescak et al. [49], comparison of the spatial contexts of distinct points can significantly improve the efficiency of the method, when compared to methods that consider intersection points alone, such as that described by Krishnamurti [50]. When comparing intersection points alone, the computational time is exponential with respect to the number of intersection points in the constructed shape. Consideration of intersection triples can significantly reduce this time, because only intersection points with the same spatial context need to be compared.

4.3 Identifying intention

Application of the subshape detection algorithm results in a list of potentially interesting parts. These are entries of the shape library which have been detected embedded in the constructed shape under Euclidean transformations. The data collected during the eye tracking studies suggests that attention to the COG of a specific part of a shape is indicative of interpretation of the shape according to that part. Accordingly, each part in the list corresponds to a potential interpretation of the constructed shape. The eye tracking interface uses the gaze of the user to determine which of the potentially interesting parts is currently recognised by the user and uses this to infer the user’s intention to manipulate the shape according to that part.

In the interface, the gaze of the user is compared to the COGs of the potentially interesting parts. If the user visually fixates on the COG of a part, then the interface infers that the user has recognised that part and intends to manipulate it. Using the Tobii X120 eye tracker, gaze data is collected at a rate of 120 Hz (i.e. every 8.33 ms), and as the user visually explores a shape this data is continuously used to calculate fixation points. Here, a fixation point is defined as the mean of a set of gaze points within a period of fixation, and is continually updated as new points are added to the set. A fixation period is defined according to a minimum duration f_d and a maximum radius f_r. A fixation starts when successive gaze points remain within a distance f_r of their mean for a duration longer than f_d, and stops when a new fixation period starts. This means that micro saccades within a fixation are ignored, since gaze points that lie further away than f_r from a fixation point for a duration less than f_d, are ignored. The values of f_d and f_r, can be specified by the user but are initially defined based on the outcomes of the eye tracking studies so that f_d = 500 ms and f_r = 50 pixels.

Figure 8 illustrates the identification of a recognised part by comparing fixation points with COGs of potentially interesting parts. Here a shape has been constructed from two rectangles, and the shape library includes a rectangle and a square which are used in subshape detection to populate the list of potentially interesting parts. The quality of the gaze data is indicated by the control in the bottom-right corner of the user interface, which here displays two white circles, indicating that data is being collected from both eyes. The position of the gaze is represented by the small circle, and is currently fixated near the COG of the square which is defined by the overlap of the two rectangles. Comparing gaze with entries in the library, the eye tracking interface infers that the user currently recognises the square, and this embedded part has been highlighted ready for selection and manipulation.

This example illustrates the possibility of using gaze data and the shape library to infer a user's current interpretation of a constructed shape, according to a recognised part. However, the fundamental limitations of eye tracking, discussed in Section 3, mean that this direct approach to visual part selection is not very effective. Firstly, there is an issue with the accuracy of the collected gaze data. If a shape is composed of parts that have COGs that are close together (within a distance of 50 pixels), then it is not possible to unambiguously determine which of the parts is being attended to based purely on gaze data. This is because the accuracy of visual fixations is limited to a radius of fixation of

approximately 50 pixels, at a viewing distance of approximately 60 cm [24]. Secondly, there is an issue with controlling the interface based on eye movements. As the user visually explores a shape, different parts are recognised from moment to moment. The eye tracking interface recognises this constant dynamic reinterpretation of shapes according to the user's visual fixations, but these typically only have duration of 500 - 600 ms. This can result in a flickering effect, with different parts of the shape highlighted in rapid succession, which is both annoying and distracting. Thirdly, using the eyes for input gives rise to the midas touch problem. This means that the interface responds to *all* eye movements of the user and if a user changes fixation, for example away from the COG of a part to one of its edges with the intention of selecting it to manipulate the part, this can result in unintended and unwanted reinterpretation of the constructed shape. To avoid these limitations, the eye tracking interface makes use of traditional mouse-based input, in addition to the gaze data and shape library. This approach is illustrated in Figure 9.

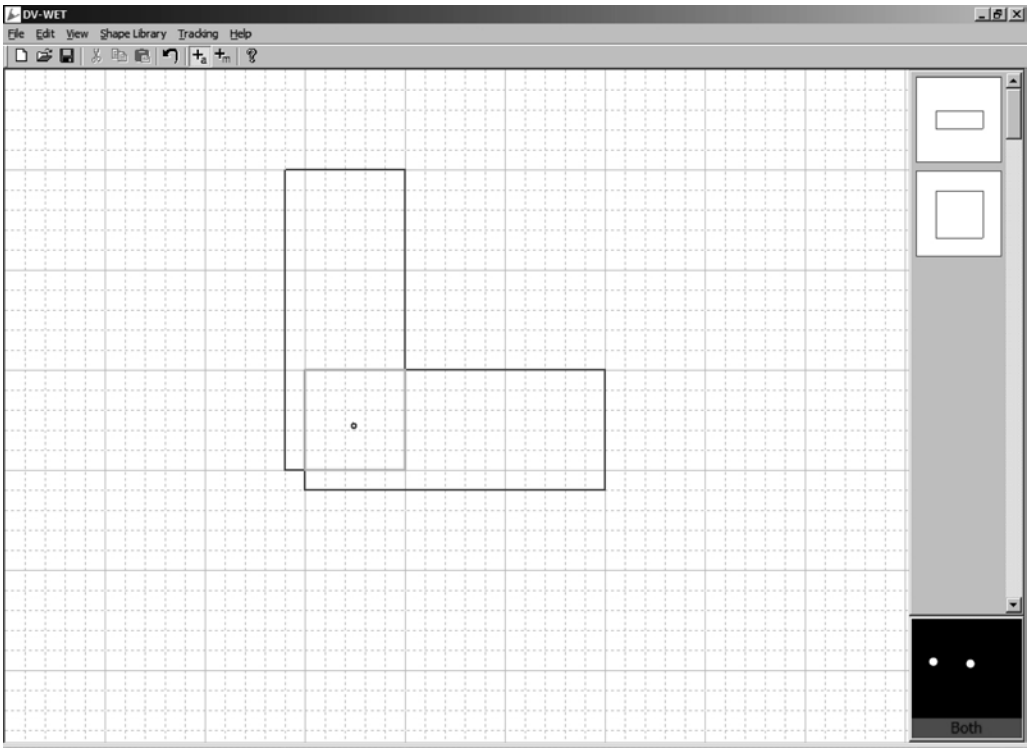

Figure 8. Subshape selection using gaze and shape library

Figure 9a illustrates purely mouse-based input, with the user selecting a line of the shape with the intention of manipulating one of its parts. However this interaction is ambiguous and it is not clear which part the user intends to manipulate. The selected

line is shared by a square and a triangle, and which of these is selected depends on how the shape was constructed, rather than the intention of the user. If the shape was constructed as four triangles, then the triangle will be selected. Alternatively if the shape was constructed as two squares, then the square will be selected. If both the triangle and the square are defined in the structure of the shape, then their order of construction will determine which is selected.

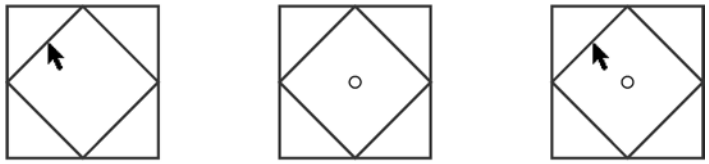

Figure 9. Identifying user's intention in shape selection

Figure 9b illustrates the gaze based approach described above, where the visual fixations are compared to the COGs of potentially interesting parts to determine which part the user intends to manipulate. Here, the position of the gaze is represented by the circle, and it is in the centre of the shape. But, this visual interaction is also ambiguous and again it is not clear which part the user intends to manipulate because the inner square and the outer square both have the same COG and the user is fixating on this point. As a result, this gaze data is not sufficient to determine which of the squares the user intends to manipulate, and instead the ordering of the list of potentially interesting parts determines which of the squares is selected.

Figure 9c illustrates the combined approach with gaze data augmenting the mouse-based input, which serves to cancel out the ambiguity in the individual approaches. Consideration of gaze data resolves ambiguity that can arise in mouse-based input, and conversely mouse-based input resolves ambiguity that can arise in data from the gaze based approach. Here, the gaze data suggests that the user intends to manipulate one of the two squares, while the mouse-based input suggests that the user is selecting either the inner square or the triangle. Therefore, a combination of the two methods of input means the intention of the user can be inferred and the user is able to select and manipulate the inner square.

This method is summarised in Algorithm 1, which is activated in the eye tracking interface in response to the user clicking the mouse on a line to select part of a constructed shape. First, the algorithm works through the list of potentially interesting

parts and compares the COGs of these with the user's current fixation point p_f. If a part in the list is found to have a COG that is within the radius of fixation f_r, of p_f then the algorithm scrolls through the lines that compose that part. If a line is found that is within a tolerance distance t of the point p_c, defined by the mouse click, then the algorithm returns the part as the part that is currently recognised by the user. If a situation arises where different parts share lines and have COGs that are within the radius of fixation f_r, of p_f, then the ordering of the list of potentially interesting parts determines which is selected.

Algorithm 1: Identification of recognised part

Input: l, list of parts; p_f, fixation point; p_c, point defined by mouse click; f_r, radius of fixation; t, tolerance
Output: *part*, fixated part

```
for (each part in l)
    cog = COG of part
    d1 = distance from cog to pf
    if (d1 < r)
        lines = set of lines defining part
        for (each line in lines)
            d2 = distance from pc to line
            if (d2 < t)
                return part
            end if
        end for
    end if
end for
```

4.4 Manipulating shape interpretations

The method summarised in Algorithm 1 identifies which part of a constructed shape has been recognised by a user at the moment when a line of the shape has been selected by a mouse click. This allows the eye tracking interface to infer which part of the shape the user intends to manipulate. But, in order to afford this manipulation the shape first needs to be reinterpreted to ensure that the part is defined within its structure. To support this reinterpretation, the eye tracking interface implements a method based on the shape grammar formalism, defined by Stiny [18].

Shape grammars are formal production systems that are defined according to shapes and shape rules. The rules are of the form $A \rightarrow B$, where A and B are both shapes, as illustrated in Figure 10a. They are used to modify shapes by recognising and replacing embedded parts under a specified set of transformations, such as the Euclidean transformations, as illustrated in Figure 10b where the rule in Figure 8a is used to rotate embedded squares. Formally, application of a rule $A \rightarrow B$ to a shape S, under a

transformation *t*, results in the modified shape (*S* – *t*(*A*)) + *t*(*B*). Shape rules can be applied repeatedly to define a sequence of shapes in a generation process, and shape grammars have previously been implemented in computational tools where design generation proceeds through processes of shape modification via rule application e.g. [22], [49].

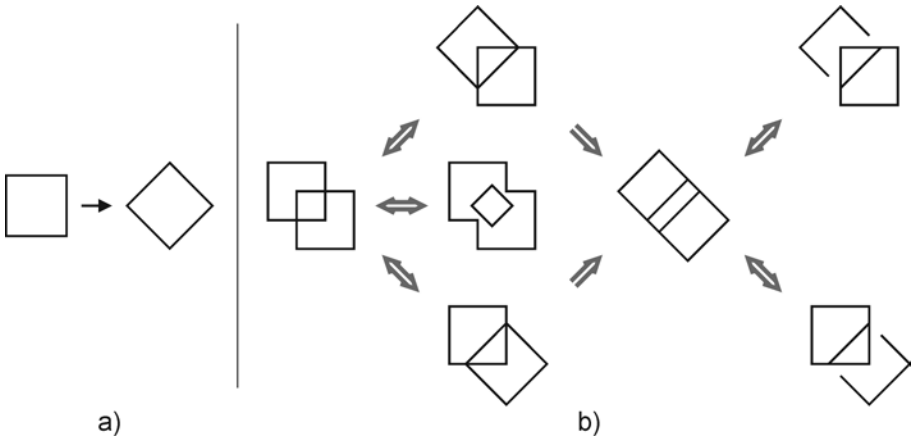

Figure 10. An example shape rule and its application

A key differentiation between shape grammars and other methods of computational generation lies in their visual nature, and this makes them particularly suitable for consideration in design research. Shape rules support manipulation of shape by identifying and replacing any part that is recognised visually embedded in the shape, including parts that emerge during a generative process [51]. Consequently, during such a process, a shape is not defined according to a fixed structure. Instead, the structure is continuously reinterpreted as a result of shape rule applications [18]. This reinterpretation mechanism is fully captured by identity shape rules, so-called *useless rules* [52]. These are shape rules of the form $A \rightarrow A$, where a shape *A* is recognised and replaced by itself, as illustrated by the rule in Figure 11a. Application of an identity shape rule to a shape S, results in the shape (*S* – *t*(*A*)) + *t*(*A*), where *t* is the transformation under which the rule is applied. In a generative process identity rules are useless because they do not modify a shape in any way: the shape (*S* – *t*(*A*)) + *t*(*A*) is visually identical to the shape *S*. But, in an explorative process they are an important observational device that supports reinterpretation of the structure of a shape according to the recognised part *A*, as illustrated in Figure 11b. Here, the identity rule in Figure 11a is applied repeatedly to recognise the different squares embedded in the initial shape in Figure 10b, and after each application of the rule the structure of the shape is redefined so that the recognised square is part of the shape.

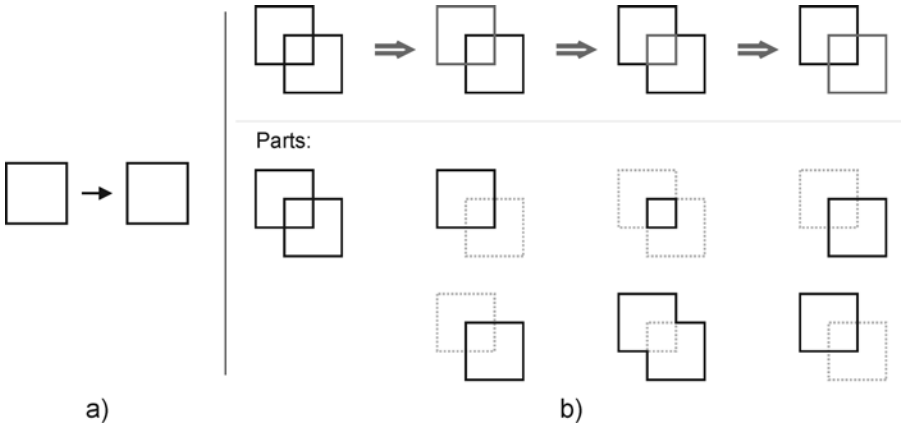

Figure 11. An example identity rule and its application

A shape rule $A \rightarrow B$ is applied according to the Boolean shape operations of shape difference and shape union, as described by Krishnamurti [50]. Given a transformation t that embeds the shape A as part of a shape S, the rule is applied by first applying the shape difference operation, $S - t(A)$. This removes the shape $t(A)$ from the shape S by removing segments of lines from the shape S that are shared by both S and $t(A)$. Next, the shape union operation $(S - t(A)) + t(B)$ is applied. This adds the shape $t(B)$ to the shape $S - t(A)$ and results in a shape that includes all the lines in $t(B)$ as well as all the lines in $S - t(A)$. When the shape rule is an identity rule the shape B is the same as the shape A, and if the shape S is reduced to its maximal representation prior to application of the rule this results in a restructuring of S into two distinct parts $t(A)$ and $S - t(A)$, as illustrated in Figure 11b.

In the eye tracking interface, when a user clicks on a line to select part of a constructed shape, the recognised part is identified according to Algorithm 1. Then, a shape identity rule $A \rightarrow A$ is defined, with the recognised part as the shape A. This rule is applied so that the constructed shape is restructured and the recognised part is defined and is available for manipulation. This is illustrated in Figure 12, where the eye tracking interface is used to support a shape exploration process. In this example, the shape library is populated with a square and a right-angled triangle, and the user has constructed a shape composed of two squares. The position of the user's gaze is represented by the small circle. In Figure 12a, the inner square is selected by the user and translated, resulting in the shape in Figure 12b. This manipulation of the shape has brought about the emergence of a triangle in the bottom-left corner of the outer square. In Figure 12c, the emergent triangle is recognised by the user which suggests that he/she intends to manipulate the shape according to this part. The eye tracking interface

is able to recognise this change in interpretation based on the combined gaze data, shape library and mouse-based input, and uses an identity shape rule to restructure the shape so that the triangle is available for manipulation, resulting in Figure 12d. Similarly, in Figure 12e, the user changes interpretation again by recognising a second emergent triangle in the top-right corner of the outer square. The intention to manipulate this triangle is again recognised and supported, and in Figure 12f the triangle has been translated.

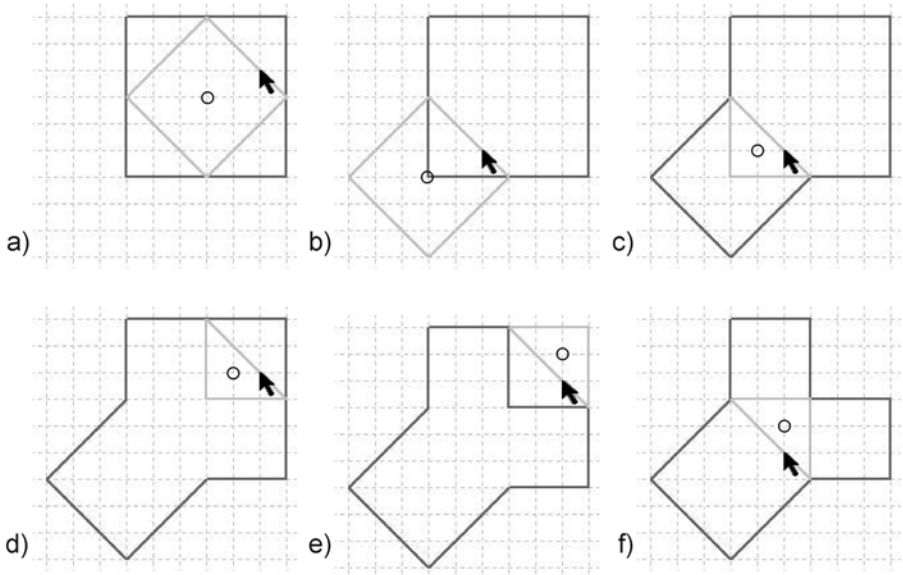

Figure 12. An example of shape exploration using the eye tracking interface

Note that, throughout this example, the user's mouse-based input is consistent and the same line is always selected at the same point. This line is an edge for the various parts that emerge throughout the shape exploration, and because the user recognises the different parts, it is possible to manipulate different interpretations of the shape. This is clearly illustrated by comparing Figures 12d and 12e, where the user recognises different triangles and consequently is able to select and manipulate those triangles in turn. At each stage of the exploration process the centres of gravity of the potential parts are used in combination with mouse-input and gaze data to determine the current interpretation of the constructed shape – if the user recognises a part that is similar to a shape in the shape library, and if a line of that part is selected using mouse-based input, then the part is made available for manipulation via definition and application of an identity shape rule. There is no additional cognitive effort required, and instead a reflective conversation can be conducted between the designer and the design representation. The designer responding to the visual structure that is apparent in the

constructed shape and, the structure of the shape is modified in response to the visual interaction of the designer.

5. Evaluating the eye tracking interface

The eye tracking interface was evaluated in a user study in which design students were tasked with various shape exploration exercises that were carried out using the two-dimensional sketch editor illustrated in Figure 2. During the study the participants used the sketch editor in different modes in order to allow familiarity with the interface and with the implemented shape exploration methods. They used the editor for traditional vector-based drawing with standard mouse-based input to construct and manipulate shapes according to defined lines. They also used the editor augmented with the shape library and shape identity rules to support reinterpretation of shapes according to recognised parts, as described in Jowers et al. [47]. Finally, they used the editor with the eye tracking interface to support dynamic reinterpretation of shapes according to recognised parts, identified by gaze data. This section is concerned with evaluation of the eye tracking interface, and as such will concentrate on the exercises in which the gaze of the participants was used to support shape exploration. These exercises were concerned with identifying how successful the eye tracking interface is at inferring interpretation of shape, according to recognised parts.

5.1 Participants

A total of ten participants were included in the user study, including six level-three product design students and four engineering design research students. All of the product design students and half of the research students had previous experience designing using CAD systems, but none of them had previous experience with eye tracking. The participants were encouraged to ask questions throughout the study to ensure that they understood the exercises and how to use the software.

5.2 Apparatus

During the user study, the software was displayed on a 17'' LCD VDU at a resolution of 1280 x 1024 pixels, with an average viewing distance of 60 cm, as illustrated in Figure 4. The participants used a traditional optical mouse as an input device, and a Tobii X120 eye tracker was used to collect the participants' gaze data – this data was used by the eye tracking interface to support reinterpretation of constructed shapes, and it was also

collected for analysis. The eye tracker was set up in a dual VDU arrangement that allowed the facilitator to monitor the participants in order to ensure that gaze data was captured as much as possible. In addition to the eye tracking data, screen capture software was use to record the participants' on-screen interactions with the shapes.

5.3 Tasks

The participants were given three exercises in which they were tasked with building complex shapes out of simple primitives. These exercises were well defined, requiring the participants to follow a series of steps to an expected conclusion, as illustrated in Figure 13. The first of the exercises, illustrated in Figure 13a, involved constructing and manipulating rectangles, and exploring the shapes that emerged. The second exercise, illustrated in Figure 13b, involved constructing two squares and manipulating them to explore the shapes that arose as a result of recognising smaller and smaller emergent squares. The third exercise, illustrated in Figure 13c, repeated this exploration for right-angled triangles.

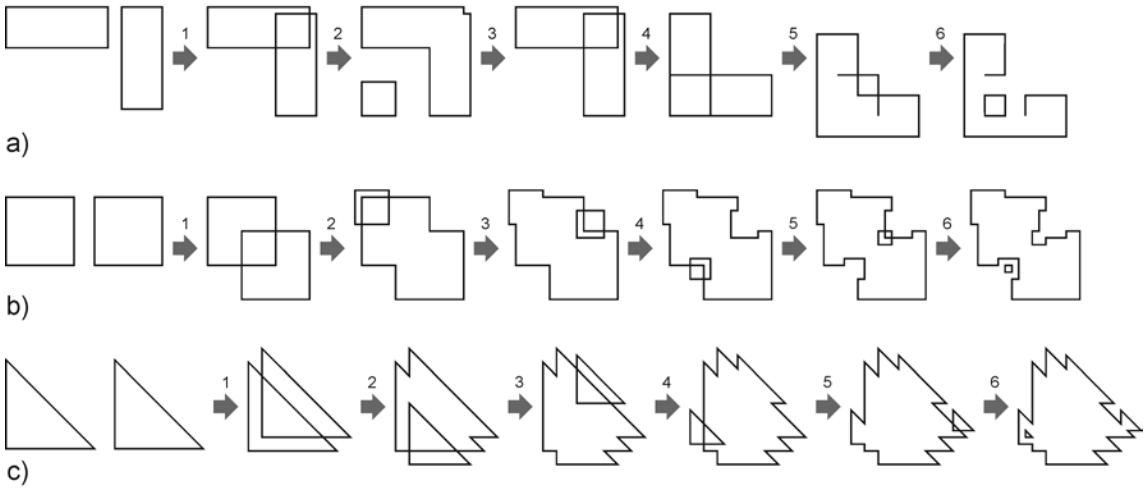

Figure 13. Shape exploration exercises

In each of the exercises a shape library was provided that was appropriate for recognising and manipulating all the parts that the participants were required to manipulate during shape exploration, and the participants used the eye tracking interface to infer their interpretation of the shapes and support manipulation of the recognised parts. There were no time restrictions, and all participants were able to complete all three of the exercises. After the participants completed the exercises they were given an opportunity to experiment using the eye tracking interface, and they were asked to complete a short survey in which they could share their thoughts and opinions

of eye tracking as an interface for CAD. The survey consisted of four questions: the first two questions asked the participants to rate how easy the eye tracking interface was to use, and how useful it was for manipulating shapes; the third question asked the participants to describe how the interface might be useful in design practice; and the fourth question asked them to describe what interactions the interface might support, that are not available in traditional CAD.

5.4 Analysis

The aim of the user study was to determine how successful the eye tracking interface is at inferring users' interpretations of constructed shapes according to recognised parts. To this end, the participants' on-screen interactions with the shapes were analysed to identify when lines were selected using mouse clicks and, in response to a mouse click, whether the interface correctly identified the current interpretation of a shape or not. If the correct interpretation was not identified then the collected data was analysed to identify why. In particular, the gaze data was analysed to identify where the user was looking at the moment a line was selected. Also the shapes were analysed according to their potential parts and the COGs of those parts, as illustrated in Figure 14. Figures 14a-c are examples from the first exercise, as illustrated in Figure 13a, and Figure 14d is an example from the third exercise, as illustrated in Figure 13c. In each of the examples, the part that is to be recognised and manipulated in the next step of the exercise is emphasised. The COGs of the potential parts are represented by circles, centred at the COG and with a radius of 50 pixels, indicating the visual tolerance defined by the specified radius of fixation. The COG of the part that is to be recognised and manipulated is represented by a filled circle.

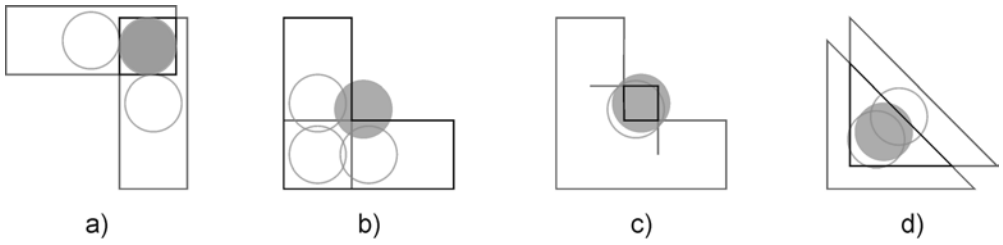

Figure 14.Examples of shapes and their potential parts from exercises in Figure 13

These four examples illustrate a range of possible outcomes, including shapes with various numbers of parts and parts with COGs that overlap by different amounts. Figure 14a illustrates Step 2 of the first exercise, where the square is to be recognised and manipulated. The shape has three potential parts, two rectangles and a square, and the

COGs of these parts touch but do not overlap. Figure 14b illustrates Step 5 of the first exercise, where the L-shape is to be recognised and manipulated. The shape has four potential parts, two rectangles, a square and an L-shape. The COG of the L-shape is external to the shape and overlaps the COGs of the rectangles by approximately 18 pixels, but does not overlap the COG of the square. Figure 14c illustrates Step 6 of the first exercise, where the square is to be recognised and manipulated. The shape has only two potential parts, a square and an L-shape, and the COGs of these overlap considerably, by approximately 85 pixels. Figure 14d, illustrates Step 2 of the third exercise, where the inner triangle is to be recognised and manipulated. The shape has three potential parts, all of which are triangles, and the COGs of these overlap considerably. The COG of the upper triangle overlaps with the COG of the inner triangle by approximately 62 pixels, while the COG of the lower triangle overlaps with the COG of the inner triangle by approximately 80 pixels.

5.5 Discussion of Results

Overall, the eye tracking interface performed reasonably well, with a success rate of 65%. This means that 65% of the time the participants were able to manipulate a shape according to the recognised parts because the eye tracking interface correctly inferred their intention as indicated by their gaze. However, this result was not consistent for all shapes, and all intended manipulations. For example, in the scenario illustrated in Figure 14a the eye tracking interface was able to successfully infer the participants' intention to manipulate the square only 50% of the time, while in the scenario illustrated in Figure 14b the interface was 76% successful at inferring the participants' intention to manipulate the L-shape. Similarly, in the scenario illustrated in Figure 14c the eye tracking interface was able to successfully infer the participants' intention to manipulate the square 85% of the time, while in the scenario illustrated in Figure 14d the interface was only 12% successful at inferring the participants' intention to manipulate the inner triangle.

Consideration of the gaze data identified reasons why the eye tracking interface was sometimes unsuccessful at inferring participants' intentions, and these are presented in Figure 15. A lack of gaze data was the reason for 37% of the failures, i.e. in these cases the participants' eyes were not detected by the eye tracker at moment when they selected a part. 21% of the failures occurred because, although the participants were fixated on the COG of the recognised part immediately prior to selecting the part, at the

moment when selection occurred (by clicking the mouse on a line of the part) they changed fixation to the mouse pointer. This is an example of the midas touch problem, where continuous eye movement can result in unwanted interaction, as discussed in Section 3. Similarly, 11% of the failures occurred because, although the participants were fixated on the COG immediately prior to selecting the recognised part, at the moment when selection occurred they changed fixation to a line or corner of the part. 21% of the failures occurred because the participants did not fixate on the COGs of recognised parts, but instead fixated on a line or corner, and 5% occurred because the gaze of the participants was not on the recognised shape prior to, or at the moment of selection.

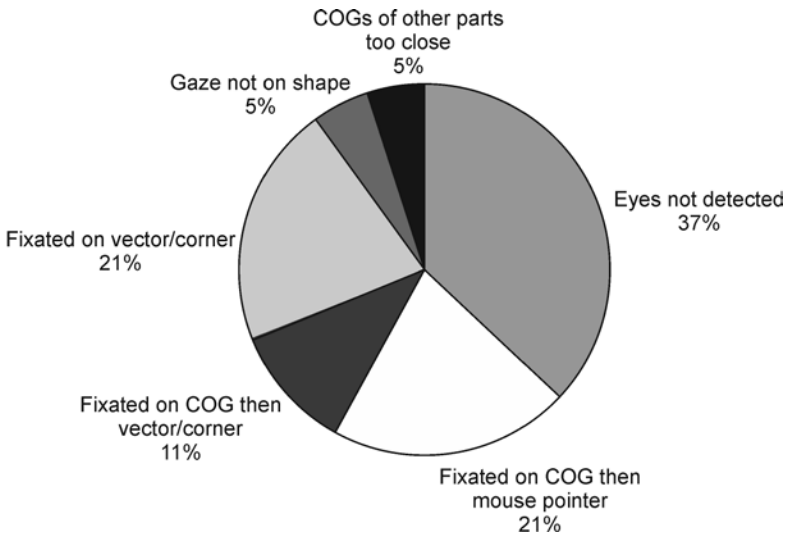

Figure 15. Reasons for eye tracking interface failure

In summary, a total 58% of the failures occurred because the participants were not fixating on the COG of the recognised part at the moment when a line of the part was selected using the mouse. Also, a total of 26% of the failures occurred because the participants did not fixate on the COG of the recognised part at all when selecting a line of the part. Only 5% of the failures occurred because the COG of the recognised part was too close to the COGs of other parts, and the reason why this isn't a bigger contribution can be attributed to the geometry of the shapes and their potential parts. In particular, the sharing of lines between parts is an important factor. This can be demonstrated by considering the two scenarios illustrated in Figures 14c and 14d. Both have considerable overlap between the COG of the recognised parts and COGs of other potential parts, but the success rate of the scenario in Figure 14c was 85% compared to a 12% success rate for the scenario in Figure 14d. This is because in Figure 14d the

recognised part, the inner triangle, shares all its lines with other potential parts that have overlapping COGs. As a result, the ambiguity of the mouse-based input and the gaze data do not serve to cancel each other out, and the method illustrated in Figure 9 is unsuccessful. This is also a contributing factor to the 50% success rate for the scenario illustrated in Figure 14a since the recognised part, the square, shares lines with the other potential parts, and although the COGs do not overlap they are close enough to cause slight ambiguity in the gaze data. On the other hand, in Figure 14c, the recognised part, the square, only shares half its lines with the other potential part, and selecting the square by clicking on the non-shared lines results in the ambiguity of the mouse-based input and the ambiguity of the gaze data cancelling each other out, and the method illustrated in Figure 9 can be applied successfully.

5.6 Summary of Findings

The success rate of the eye tracking interface in inferring participants' interpretations of shapes according to their parts is high enough to suggest that the methods used in implementing the interface are appropriate. For example, the data collected during the user study strongly supports the results from the previous eye tracking studies, as described in Section 4.1, and confirms that fixation on the COG of a part of a shape is indicative of interpretation of the shape according to that part, as illustrated in Figure 16. In 78% of the cases the participants looked at the COG of the recognised part during interpretation of the shapes. However, in 11% of the cases the participants looked away at the moment they selected the part and in 2% of the cases the COG was close to the COGs of other potential parts, resulting in unsuccessful inference of the participants' intentions. In 13% of the cases there was a lack of gaze data. In only 9% of the cases did the participants not look at the COG of the recognised part.

The combination of gaze data, mouse-input and the shape library provided an intuitive and dynamic approach to shape exploration that the participants of the user study appreciated. This was indicated in the participants' responses to the survey. Most of the participants found the eye tracking interface easy to use, and found it useful for manipulating shapes. When asked how the interface might be useful in design practice, the participants responded by saying that it could make interaction with digital shapes faster and easier, e.g. *"it would make drawing shapes quicker than by hand"*. When asked what interactions the interface might support that are not available in traditional CAD they recognised that gaze data was a suitable interface for selecting, e.g. selecting

aspects of geometry such as parts, faces, etc. or selecting options or properties such as colour. One participant also recognised that eye tracking has potential benefit to support collaborative design, with gaze data used to make the intentions of a designer more apparent to his/her colleagues.

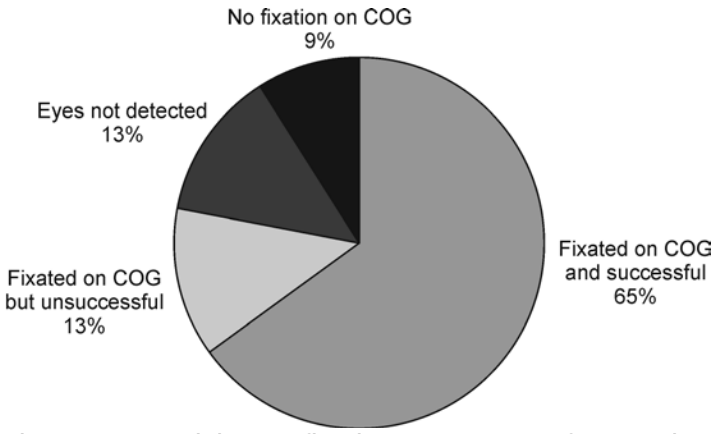

Figure 16. Participants fixations on COGs of recognised parts

The user study also highlighted limitations of the eye tracking interface. In particular, the midas touch problem is still an issue, despite using a combination of gaze and mouse-based input to try and circumvent it. In 11% of the cases of shape interpretation, participants looked away at the moment they selected a recognised part despite fixating on the COG of the part prior to selecting it. This is a typical example of the midas touch problem, where it's not possible to distinguish between gaze that is intended as interaction, and gaze that results from collecting visual information. In the eye tracking interface gaze is not explicitly used to interact with shapes, instead the natural eye movements of the user are used to infer intention. As a result, a mode switching solution such as that described by Istance et al. [40] is unlikely to be appropriate, and it is not clear how to address the problem further. However, as discussed in Section 3, midas touch is a fundamental problem in any eye tracking application and continuing research explores new methods to circumvent it. This research should be considered in future developments of the eye tracking interface.

6. Discussion

Gaze is an implicit indicator of attention and intention, and eye tracking has significant potential as an additional interface that augments traditional mouse and keyboard by responding to users' visual interactions. With respect to design, eye tracking can be

used to substantially reduce the cognitive overhead needed for designers to interface with CAD systems by inferring their intended manipulations of design models. The prototype eye tracking interface presented here is a first imagining of the possibilities, and explores how gaze data can be used to infer users' interpretation of ambiguous shapes during shape exploration. Evaluation of the interface was positive, and throughout the user study it was successfully able to respond to participants' visual intentions, allowing manipulation of recognised parts of shapes. The participants also commented that the interface could be a useful addition to CAD, supporting faster and easier form creation.

Shape interpretation is an essential process in design exploration, but is not readily supported in commercial CAD systems. As a result, Robertson and Radcliffe [3] note that use of CAD can result in premature fixation because of a reluctance to incorporate ideas that will result in too many changes to a constructed design model or its underlying structure. Fundamentally, this issue manifests because digital design models impose a distance between designers and the representations that they work with, a distance which does not arise with physical design media, such as sketches [15]. In sketches, there is no distinction between the visual shape and its representation. The shape *is* the shape, and this means that visual interpretations can immediately be incorporated into shape exploration. Conversely, when designers work with computational tools such as CAD, the visual shape is a rendition of formal data structures which have a single interpretation. The shape is *not* the shape but is a visualisation of a particular data structure – the ambiguity and richness of the visual shape are not accounted for. The eye tracking interface presented here tries to address this problem by supporting dynamic reinterpretation of shapes according to the visual structure that is recognised by the designer from moment to moment. As such it supports explorative processes, with the designer free to interpret shapes according to dynamically changing structure, thereby encouraging what Schön and Wiggins [17] refer to as the *reflective conversation* between the designer and the design representation. Also, because the interface supports multiple interpretations of shape it affords ambiguity, uncertainty and parallel lines of thought, which Lawson and Locke [4] argue are central to creative design. Eye tracking can respond to visual interaction with digital design models and compliment design processes, by literally supporting the seeing-moving-seeing process of creative shape exploration. This suggests that an eye-tracking interface could be used to provide

more appropriate computational support during conceptual design and could result in a computerised design process that encourages creative design practice. However these claims have not been tested, but instead are an area for investigation in future user studies.

The prototype is only one of many potential applications for eye tracking as an interface for CAD. As identified by the participants in the user study, eye tracking can serve as a useful, implicit, method for selection and could, for example, be used to simplify navigation of the complex menu systems that are common in CAD systems. Or, as discussed in the previous section, it could also be useful in collaborative design, to make the intentions of members of design teams apparent to colleagues. Also, with respect to shape, gaze could be a useful guide for localizing mouse-based input on 3D digital models, enabling more efficient manipulation. The rapidly diminishing cost of eye tracking technology, commercially and as a result of open hardware projects such as openEyes [53], mean that eye tracking interfaces are becoming a realistic possibility for future human-computer interaction. This gives rise to exciting possibilities for designers, enabling them to control their computational tools using methods previously not possible.

Acknowledgements

The research reported in this paper was carried out as part of the Designing with Vision project, and the prototype described is available on the project website[3]. The Leverhulme Trust funded this research but had no further involvement. The authors would like to thank the University of Leeds students who participated in the user studies.

References

[1] B. Lawson, Oracles, draughtsmen, and agents: the nature of knowledge and creativity in design and the role of IT, Autom. Constr. 14 (2005) 383-391.
[2] J.F. Blinn, The ultimate design tool, Comput. Graph. App. 10 (1990) 90-92.
[3] B.F. Robertson, D.F. Radcliffe, Impact of CAD tools on creative problem solving in engineering design, Comp.-Aided Des., 41 (2009) 136-146.
[4] B. Lawson, S.M. Loke, Computers words and pictures, Des.Stud. 18 (1997) 171-183
[5] R. Hanna, T. Barber, An inquiry into computers in design: attitudes *before*–attitudes *after*, Des. Stud., 22 (2001) 255-281.
[6] M. Stacey, C. Eckert, CAD system bias in engineering design, in: International Conference on Engineering Design, Technical University of Munich, 1999, pp. 1413-1418.

[3] http://design.open.ac.uk/DV

[7] W.J. Mitchell, Vitruvius Redux, in: E.K. Antonsson, J. Cagan (Eds.) Formal Engineering Design Synthesis, Cambridge University Press, Cambridge, 2001, pp. 1-19.
[8] B.L. Jenkins, Making CAD easier proves challenging, Comp. Graph. World, 21 (1998) 21-22.
[9] B. Sener, P. Wormald, User evaluation of HCI concepts for defining product form, Des. Stud., 29 (2008) 12-29.
[10] Sutherland, I. Sketchpad: A man-machine graphical communication syste, PhD Thesis, Massachusetts Institute of Technology, 1963.
[11] J.H. Israel, E. Wiese, M. Mateescu, R. Stark, Investigating three-dimensional sketching for early conceptual design - Results from expert discussions and user studies, Comp. and Graph., 33 (2009) 462-473.
[12] S. Abdelmohsen, E. Do, TangiCAD: Tangible Interface for Manipulating Architectural 3D Models, in: 12th International Conference on Computer Aided Architectural Design Research in Asia, CAADRIA, 2007, pp. 29-36.
[13] A. Duchowski, Eye Tracking Methodology: Theory and Practice, 2nd ed., Springer, London, 2007.
[14] E. Saund, T. Moran, A perceptually supported sketch editor, in: Symposium on User Interface Software and Technology, ACM, 1994, pp. 175-184.
[15] I. Jowers, C. Earl, Shape interpretation with design computing, in: J.S. Gero (Ed.) International Conference on Design Computing and Cognition, Springer, 2012.
[16] M. Suwa, Constructive perception: Coordinating perception and conception toward acts of problem-finding in a creative experience, Jpn. Psychol. Res., 45 (2003) 221-234.
[17] D.A. Schön, G. Wiggins, Kinds of seeing and their functions in designing, Des. Stud., 13 (1992) 135-156.
[18] G. Stiny, Shape: Talking about Seeing and Doing, MIT Press, Cambridge, 2006.
[19] M.D. Gross, Emergence in a recognition based drawing interface, in: J.S. Gero, B. Tversky, T. Purcell (Eds.) Visual and Spatial Reasoning in Design II, University of Sydney, 2001, pp. 51-65.
[20] Y.-T. Liu, Some phenomena of seeing shapes in design, Des. Stud., 16 (1995) 367-385.
[21] R.B. Tilove, A.A.G. Requicha, Closure of Boolean operations on geometric entities, Comp.-Aided Des., 12 (1980) 219-220.
[22] I. Jowers, D.C. Hogg, A. McKay, H. H.Chau, A.d. Pennington, Shape detection with vision: Implementing shape grammars in conceptual design, Res. Eng. Des., 21 (2010) 235-247.
[23] D.H. Hubel, Eye, Brain and Vision, New Edition ed., Henry Holt and Co., 1995.
[24] R.H.S. Carpenter, Movements of the Eyes, Pion, London, 1977.
[25] M.I. Posner, Orienting of attention, Q. J. Exp.Psychol., 32 (1980) 3-25.
[26] J.E. Hoffman, B. Subramaniam, The role of visual attention in saccadic eye movements, Percept. Psychophys., 57 (1995) 787-795.
[27] A.L. Yarbus, Eye Movements and Vision, Plenum, 1967.
[28] R. Pieters, A Review of Eye-Tracking Research in Marketing, in: N.K. Malhotra (Ed.) Review of Marketing Research, Emerald Group Publishing Limited, 2008, pp. 123-147.
[29] R.J.K. Jacob, K.S. Karn, Eye tracking in Human-Computer Interaction and usability research: Ready to deliver the promise, in: J. Hyönä, R. Radach, H. Deubel (Eds.) The mind's eye: Cognitive and applied aspects of eye movement research, Elsevier, Amsterdam, 2003, pp. 573-605.
[30] R.M. Wilkie, J.P. Wann, Eye-Movements Aid the Control of Locomotion, J. Vis., 3 (2003) 677-684.
[31] M. Porta, Vision-based user interfaces: methods and application, Int. J. Hum.-Comp. Stud., 57 (2002) 27-73.
[32] M. Kumar, A. Paepcke, T. Winograd, EyePoint: Practical Pointing and Selection Using Gaze and Keyboard, in: Conference on Human Factors in Computing Systems, ACM, 2007, pp. 421-430.
[33] P. Majaranta, K.-J. Räihä, Twenty Years of Eye Typing: Systems and Design Issues, in: Symposium on Eye Tracking Research & Applications, ACM, 2002, pp. 15-22.
[34] J.O. Wobbrock, J. Rubinstein, M.W. Sawyer, A.T. Duchowski, Longitudinal evaluation of discrete consecutive gaze gestures for text entry, in: Symposium on Eye Tracking Research & Applications, ACM, 2008, pp. 11-18.

[35] A.J. Hornof, A. Cavender, EyeDraw: enabling children with severe motor impairments to draw with their eyes, in: Conference on Human Factors in Computing Systems, ACM, 2005, pp. 161-170.
[36] J. Gips, P. DiMattia, F.X. Curran, P. Olivieri, Using EagleEyes-an electrodes based device for controlling the computer with your eyes-to help people with special needs, in: International Conference on Computers Helping People with Special Needs, Oldenbourg Verlag, 1996, pp. 77-83.
[37] R. Bates, H.O. Istance, Why are eye mice unpopular? A detailed comparison of head and eye controlled assistive technology pointing devices, Univers. Access Inf. Soc., 2 (2003) 280-290.
[38] R.J.K. Jacob, Eye Movement-Based Human-Computer Interaction Techniques: Towards Non-Command Interfaces, in: H.R. Hartson, D. Hix (Eds.) Advances in Human-Computer Interaction, Ablex Publishing Co., Norwood, 1993.
[39] R. Bates, H. Istance, Zooming Interfaces! Enhancing the Performance of Eye Controlled Pointing Devices, in: International Conference on Assistive Technologies, ACM, 2002, pp. 119-126.
[40] H. Istance, R. Bates, A. Hyrskykari, S. Vickers, Snap Clutch, a Moded Approach to Solving the Midas Touch Problem, in: Symposium on Eye tracking Research & Applications, ACM, 2008, pp. 221-228.
[41] R.A. Bolt, Gaze-orchestrated Dynamic Windows, Comp. Graph. World, 15 (1981) 109-119.
[42] J.D. Smith, T.C.N. Graham, Use of eye movements for video game control, in: International Conference on Advances in Computer Entertainment Technology, ACM, 2006.
[43] S.M. Yoon, H. Grag, Eye Tracking based Interaction with 3D Reconstructed Objects, in: International Conference on Multimedia, ACM, 2008, pp. 841-844.
[44] L.E. Sibert, R.J.K. Jacob, Evaluation of Eye Gaze Interaction, in: Conference on Human Factors in Computing Systems, ACM, 2000, pp. 281-288.
[45] Tobii Eye Tracking: An Introduction to eye tracking and Tobii eye-trackers, White Paper, from: http://www.tobii.com/Global/Analysis/Training/ WhitePapers/Tobii_EyeTracking_Introduction_WhitePaper.pdf, accessed on 2/8/2012, Tobii Technology, 2010.
[46] M. Prats, I. Jowers, N. Pedreira, S. Garner, A. McKay, Interpretation of geometric shapes: an eye movement study, in: Symposium on Eye-Tracking Research & Applications, ACM, 2010, pp. 243-250.
[47] I. Jowers, M. Prats, A. McKay, S. Garner, Design exploration with useless rules and eye tracking, in: S.J. Culley, B.J. Hicks, T.C. McAloone, T.J. Howard, A. Dong (Eds.) International Conference on Engineering Design, The Design Society, 2011, pp. 443-455. [48] D. Vishwanath, E. Kowler, Localization of shapes: eye movements and perception compared, Vis. Res., 43 (2003) 1637-1653.
[49] T. Trescak, M. Esteva, I. Rodriguez, A shape grammar interpreter for rectilinear forms, Comp.-Aided Des., 44 (2012) 657-670.
[50] R. Krishnamurti, The construction of shapes, Environ. Plan. B-Plan. Des., 8 (1981) 5-40.
[51] T. Knight, Computing with Emergence, Environ. Plan. B-Plan. Des., 30 (2003) 125-155.
[52] G. Stiny, Useless rules, Environ. Plan. B-Plan. Des., 23 (1996) 235-237.
[53] D. Li, J. Babcock, D.J. Parkhurst, openEyes: a low-cost head-mounted eye-tracking solution, in: Symposium on Eye Tracking Research & Applications, ACM, 2006, pp. 95-100.