Jokinen, Jussi P.P.; Wang, Zhenxin; Sarcar, Sayan; Oulasvirta, Antti; Ren, Xiangshi

Adaptive feature guidance: Modelling visual search with graphical layouts

Published in:
International Journal of Human Computer Studies

DOI:
10.1016/j.ijhcs.2019.102376

Published: 01/04/2020

Document Version
Publisher's PDF, also known as Version of record

International Journal of Human-Computer Studies 136 (2020) 102376

Contents lists available at ScienceDirect

International Journal of Human-Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

Adaptive feature guidance: Modelling visual search with graphical layouts

Jussi P.P. Jokinen[*,a], Zhenxin Wang[b], Sayan Sarcar[b,c], Antti Oulasvirta[a,d], Xiangshi Ren[b]

[a] Department of Communications and Networking, Aalto University, Finland
[b] Center for Human-Engaged Computing (CHEC), School of Information at Kochi University of Technology, Japan
[c] Faculty of Library, Information and Media Science, University of Tsukuba, Japan
[d] Finnish Center for Artificial Intelligence FCAI

ARTICLE INFO

Keywords:
Visual search
Computational modelling
Learning

ABSTRACT

We present a computational model of visual search on graphical layouts. It assumes that the visual system is maximising expected utility when choosing where to fixate next. Three utility estimates are available for each visual search target: one by unguided perception only, and two, where perception is guided by long-term memory (location or visual feature). The system is adaptive, starting to rely more upon long-term memory when its estimates improve with experience. However, it needs to relapse back to perception-guided search if the layout changes. The model provides a tool for practitioners to evaluate how easy it is to find an item for a novice or an expert, and what happens if a layout is changed. The model suggests, for example, that (1) layouts that are visually homogeneous are harder to learn and more vulnerable to changes, (2) elements that are visually salient are easier to search and more robust to changes, and (3) moving a non-salient element far away from original location is particularly damaging. The model provided a good match with human data in a study with realistic graphical layouts.

1. Introduction

This paper investigates a central topic in human-computer interaction (HCI) research: how users learn graphical user interfaces and how changes in visual design affect performance (Chen and Liu, 2008; Cockburn et al., 2007; 2015; Dunlop and Levine, 2012; Ehret, 2002; Jokinen et al., 2017; Paik et al., 2015; Sears et al., 2001; Somberg, 1987). People are continuously exposed to new, updated, or redesigned interfaces. When making decisions on whether to continue using a changed interface they weigh the expected benefits against costs such as learning costs (Venkatesh et al., 2016). A change in a familiar design can have an instant effect on usability and can cause frustration. This paper contributes to the understanding of key questions in this space: (1) why some changes are less costly to learn, and (2) why some layouts are more robust to change.

Our long-term goal is an accurate and plausible model to support decision-making and design (Bailly et al., 2014; Byrne, 2001; Card et al., 1983; Cockburn et al., 2007; John et al., 2004; Kieras and Hornof, 2014; Kieras and Meyer, 1997). Although computational models have been successful in predicting expert performance in a range of interactive tasks, much work remains to be done in modelling how novices gradually become more skilled users (Jokinen et al., 2017). Novice-to-expert transition is a central problem in HCI, and the psychological understanding of this transition will help to design interventions that aid users in becoming experts (e.g., Malacria et al., 2013). The importance of such models is in their capacity to entertain counterfactual scenarios, allowing the modeller to change the task and user parameters to investigate various types of "what ifs". This is necessary for interface optimisation (e.g., Sarcar et al., 2018) and research of psychological basis of interface use (e.g., Borst et al., 2015).

This paper contributes to models of visual search and learning in the domain of graphical interfaces. The model builds on existing work in the modelling of visual attention and learning in HCI. It covers a large number of known, important visual primitives of regular graphical layouts, including element locations (Ehret, 2002; Jokinen et al., 2017) as well as visual features like size and shape and color (Ehret, 2002; Nyamsuren and Taatgen, 2013), alongside with relearning of changed layouts (Jokinen et al., 2017). The model predicts how users (1) visually search layouts, accounting for the top-down and bottom-up effects of the visual features (e.g., colour, size, and shape) of the layout elements but also the number of these elements; (2) learn the visual locations and features of the elements over time; and (3) relearn layouts that have been changed. There exist multiple models for predicting one or two of these three considerations (Ehret, 2002; Jokinen et al., 2017; Kieras and Hornof, 2014), but so far none has covered all three. The new model can simulate how users cope with a wide array of real-life

* Corresponding author.
E-mail address: jussi.jokinen@aalto.fi (J.P.P. Jokinen).

https://doi.org/10.1016/j.ijhcs.2019.102376
Received 28 February 2019; Received in revised form 26 July 2019; Accepted 12 November 2019
Available online 30 November 2019

layouts, ranging from web sites and menus to mobile UIs.

Our point of departure is the observation that human vision is fundamentally limited in its information-processing capacity (Cave and Bichot, 1999; Rayner, 2009; Salvucci, 2001). Firstly, the anatomy of the eye restricts the region which can be seen clearly. Visual acuity is best in the fovea, subtending 2° from the centre of vision (*fixation*), and worsens quickly towards the parafovea (5°) and peripheral region (beyond parafovea). This necessitates ballistic eye movements (*saccades*), which bring the fixation and thus visual clarity to areas of interest. Preparing and executing such a movement takes time, and this again limits the information processing capacity of the human vision. These bounds result in the emphasis of *strategic adaptation* in determining how to search for a target. In other words, the visual system is confronted with *a sampling problem*: it has to decide where to deploy attention next (Wolfe and Horowitz, 2017). To approach this problem, we assume that the visual system is "cognitively bounded rational" or "computationally rational" (Gershman et al., 2015; Lewis et al., 2014). The control of gaze in a visual search is understood as utility-maximisation under constraints posed by the task, the capabilities of the visual system, and the layout.

To model the adaptive capability in visual search, our model converges psychological principles under a concept we call *adaptive feature guidance*. It instantiates three cognitive hypotheses about learning, memory, and vision to model visual search with UIs. The main principles converging in the model are *utility learning* (Anderson, 2007; Rescorla and Wagner, 1972), *associative learning* (Anderson, 1976; Anderson et al., 1998) and *feature guidance* (Nyamsuren and Taatgen, 2013; Wolfe and Horowitz, 2017). We assume that a *controller* chooses a set of features from two sources of information so as to maximise the expected learned utility of the next attention shift: bottom-up and top-down. On one hand, every visual object in a layout has a saliency value determined by its visual features, such as size, shape, and colour, and by its position with respect to focal and peripheral vision and to surrounding visual elements (Chen et al., 2015; Kieras and Hornof, 2014; Nyamsuren and Taatgen, 2013; Treisman and Gelade, 1980; Williams, 1967; Wolfe and Horowitz, 2017). On the other hand, its choice can be informed by two memory systems (short- and long-term). The visual short-term memory (VSTM) tries to inhibit visits to already visited locations. It serves as an efficient means of supporting visual search when the layout is new. Moreover, as the model searches for targets, information about them is encoded within associative long-term memory (LTM) storage. This stores associations between objects and their features (Anderson, 1976; Anderson et al., 1998). With more exposure to a layout, the controller learns to rely more on LTM, allowing it to use feature guidance of vision towards targets sooner. This memory, however, does not learn perfectly and is compromised by decay over time.

In sum, the adaptive feature guidance model simulates visual search, which has adaptive resources (utility learning, STM, LTM) to guide the search with an emphasis on the features of the visual elements. The model's adaptive nature means that the search policy, that is, the actions that the model takes in its search through a UI, is generated and updated adaptively. For instance, assume a model that simulates expert behaviour in that it recalls the positions of all UI elements, and a sudden modification of the layout such that some of the elements change position but remain visually the same. The model learns to not utilise its positional knowledge of the moved elements, instead relying on what it remembers about their visual features to guide its search, until it relearns the new element locations.

The model's scope is graphical interfaces. Compared to the totality of visual scenes we deal with, graphical UIs involve a fairly standardised way of representing the visual elements and their features. They represent a computer program visuospatially for commanding with a pointing device. The state and controls of a computer program are represented as non-overlapping spatially bounded objects (e.g., rectangular areas such as buttons) that permit interactions that change their state. Our model takes as input a layout with target elements and optionally their frequencies as encountered in the past. It then simulates visual search for these targets and outputs eye-movement data and search time. The output can be used for detailed analysis of single-task eye-movement behaviour, or the data can be aggregated to produce average search times. In addition, the layout can be changed mid-run to simulate relearning.

The model simulates visual learning of layouts over repeated trials. Initially, the model has no prior knowledge of the layout and search is slow, except for visually distinct elements. As the model starts to learn by creating associations of targets with their locations and visual features, search becomes faster. If the layout changes, the model cannot utilise its memory of the element locations and must relearn them. However, if the elements retain their distinct features, the model uses its knowledge concerning these features to quickly find the moved element. Due to utility learning, the model adapts to the utility of the available resources. The better the information provided by a resource, the more it is relied upon. If location information is no longer useful due to a changed layout, the model prefers to use feature information instead in guiding attention, until the location information has been updated to a level where using it is again the most efficient search strategy.

Practitioners can use the model as an evaluation tool and in computational design. The input consists of: (1) a segmented a layout, consisting of rectangular elements with color and size properties; (2) the target element and (3) the user's prior history with the layout (if any). Three types of predictions can be given to answer the following questions.

- *Search time:*Given a layout with a number of graphical elements and their locations and features, what is the visual search time for a given element?
- *Learning time:*How long does it take to learn a new layout, given the number, locations, and visual features of elements?
- *Effects of changing a layout:*Given a user history with a layout and a new layout that is a variation thereof, what is the initial impact of the layout change on visual search times, and how long does it take to relearn the changed layout?

Predictions to these questions help designers in some recurring problems. First, a designer can evaluate candidate designs. The model can predict how difficult a new layout is to learn, either on its own or compared to a previous layout with which users have experience. Competing designs can be compared similarly. Second, a designer can apply computational optimisation methods, such as combinatorial optimisation, to search through a larger space of possible designs more systematically. Here the model serves as an objective function for the optimiser. Third, a designer may want to assess how to best change a layout, for example when removing or adding new features. The model allows making changes to the layout during simulations. We demonstrate some of these uses at the end of the paper.

The present model deals with single visual targets that the simulated user wishes to find as quickly as possible. This excludes modelling of multistage tasks, where the user needs to find information from multiple visual elements during the same task. Also excluded are dynamic tasks, where the interface changes during the visual search (but our model *does* predict the impact of interface changes that occur between search tasks). In addition, the model is limited to visual searching, and thus does not simulate actual interaction with the UI, such as mouse clicks or button presses. Further, the interfaces presented to the model are defined symbolically as objects. Any analytics on images of UIs need to go through a separate segmentation phase to be translated into model-runnable representations.

In the following, we first describe the model and the principles behind it, supporting them with a detailed walkthrough examples. We then present results from an experiment testing the validity across a range of realistic layouts: a website, a consumer interface, and an

operating system UI. We use the model to investigate various practical problems related to layout learning, such as the effect of a new salient layout element (e.g. an advertisement) on a visual search or a grouping of items in a layout by their visual features. We conclude with a discussion of design implications and future work.

2. Related work and goals

2.1. Models of vision and learning

Learning of UIs has been a topic of long-term interest in HCI research (Anderson and Bischof, 2013; Bederson et al., 2015; Chen and Liu, 2008; Chen et al., 2017; Keating et al., 2016; Kosmyna et al., 2015; Paik et al., 2015; Rieman, 1996). However, only a few papers have looked at the general case of learning graphical user interfaces. Fleetwood and Byrne (2006) examined visual search strategies for UIs, with a close focus on eye movements. However, their model did not cover the effect of learning on eye-movement patterns and search times. A recent paper presented a model of layout learning, but it focused on keyboards only (Jokinen et al., 2017). Keyboard layouts are a constrained instance of graphical layouts wherein elements are presented in a grid lattice. All elements are further assumed to be of the same size, colour, and shape. Conversely, a model utilising an *active vision* approach emphasised the features of visual elements and also included the notion of visual threshold or acuity with respect to the features (Kieras and Hornof, 2014). However, this model did not cover learning. An earlier model of layout learning utilised position learning and addressed some aspects of feature-based search but did not explicate a model of full feature guidance with visual threshold, nor did it model relearning of changed layouts (Ehret, 2002). The proposed model combines ideas from these three and other modelling related research, in order to create a full model of layout learning. To this end, models of vision and learning are here reviewed.

2.1.1. Vision

A model of visual search on graphical UIs must make realistic predictions about the eye movements of the users. Due to visual processing constraints, the user does not usually see the all visual UI elements at once and in detail, but instead can encode only a *foveated* subset of the full UI. This means that even if the entire layout is shown on one screen, the user is not able to recognise all of its menus, icons, fields, and other elements. Therefore, the main problem in visually searching graphical UIs becomes the problem of attention deployment: where to look next? This involves both modelling where the visual attention of the user is deployed, given the visual elements of the UI, as well as what the results of this attention deployment are - namely in terms of eye movements and visual encoding of the attended targets.

Many recent visual search models have utilised the EMMA eye-movement model (Salvucci, 2001) for simulating how the eyes move in various tasks, such as reading (Salvucci, 2001), visual search (Fleetwood and Byrne, 2006; Jokinen et al., 2017), and driving (Kujala and Salvucci, 2015; Salvucci and Macuga, 2002). Likewise, the adaptive feature guidance model described in this paper utilises the EMMA model. EMMA describes the cognitive control process of attending and encoding visual elements. It models visual encoding of elements as both unobserved attention shifts and observable eye movements. The model predicts that not all attention shifts require eye movements, because people can also encode elements close to their current fixation within the limits of visual acuity. While EMMA predicts eye movements and encoding time, it does not make statements about what the encoded target should be.

The problem of eye-movement targeting or attention deployment has been investigated using different types of computational models (Kowler, 2011). Usually these models employ the ideas of *bottom-up* and *top-down* processes, either individually or together (Tatler et al., 2005). The bottom-up process refers to the finding that *salience* of a visual region is a major determinant of attention deployment. Models utilising the notion of a bottom-up process can be called *map-based*. They compute salience maps from input images, highlighting regions that are visually conspicuous. One popular computational implementation of this approach is the search model by Itti and Koch (2000). It assumes that visual attention is directed in a winner-take-all fashion to the visual region that is most conspicuous, that is, that which stands out the most from its surroundings. In order to prevent the most salient region being the only area under attention, the model implements an *inhibition of return*, where recently attended regions temporarily lose strength. The assumptions of visual salience, winner-take-all attention deployment, and inhibition of return are also present in our model of visual search.

Top-down processes depict the role of the high-level visual search task in determining the region that is attended to next. This means that attention deployment depends on the search task. A top-down process requires that the agent, such as a user of a visual UI, has task-relevant knowledge about the features of the desired targets. This *top-down feature guidance* can be in the form of visual features, such as colour, orientation, and size (Wolfe and Horowitz, 2017), or in the form of explicit knowledge about the location of the target (Jokinen et al., 2017). The bottom-up and top-down models of visual attention employ two different sources for attention deployment, salience and guidance, but the approaches can be integrated into a componential visual search model, that allows the modelling of both propositions.

An example of a bottom-up and top-down componential model of visual search is *PAAV* (Nyamsuren and Taatgen, 2013). It describes the human visual system as a representation of the environment, where visual acuity thresholds for individual features, such as their colour, shape, and size, affect how the environment is represented. As such, it is modelled after the computational *guided search model* (Wolfe, 1994). It distinguishes between two stages of visual processing: pre-attentive and attentive. The pre-attentive process collects automatically and in parallel fashion information about the elements of the visual world, such as their locations and visual features, as long as these elements are within the visual acuity thresholds, given the current fixation location, and can thus be detected by the automatic visual system. This process is used to calculate the *bottom-up activation* of the visual items in the environment. In addition to the bottom-up pre-attentive stage, the model has a top-down attentive stage, which is used to guide attention towards elements that have *top-down activation*, that is, similarity to any visual features requested by the model. In sum, the model presents a set of formulas that are used to calculate bottom-up and top-down activations of an element from its features that are visible in light of the current fixation, and states that attention is deployed to the visual element with the largest weighted sum of these two sources of activation. A similar description of vision is present in a model called *active vision* (Kieras and Hornof, 2014). However, that model simplifies the attention deployment and does not utilise a combination of low-level bottom-up and top-down saliency calculation. In the case of the visual search of layouts, both element saliency and top-down guidance should be accounted for.

2.1.2. Learning

There is a large body of work on the modelling of skill and skill acquisition (e.g., Anderson, 1976; Anderson, 2007; Anderson, 2013; Anderson et al., 1998; Janssen and Gray, 2012; Newell, 1990; Veksler et al., 2014). Learning models can be grouped into descriptive (statistical) and cognitive models. A classic example of a *descriptive aggregated model* is the *power law of learning*, which predicts how repeated exposure improves performance Newell and Rosenbloom (1981). Initially, the performance improves rapidly, but the rate of performance improvement decreases quickly as the function of repetitions. This leads to an asymptotic learning curve, where the expert performance settles close to the asymptotic. The fitting of the three parameters of the power function allows for the description of the learning curve and its asymptotic. Although the function itself is simple, it has been shown to

approximate learning in different scenarios from simple perceptual-motor skills to learning geometric proofs (Newell and Rosenbloom, 1981).

The benefit of descriptive models, such as the power law of learning, is their relatively simple composition, which results in clear predictions and short computation times. However, these models do not make hypotheses about the mechanisms of learning, and cannot be used to predict learning outside the context in which they have been fitted. Therefore, it is difficult to use them in counterfactual reasoning, and they are not robust against changes in the task without parameter re-fitting. For example, if we were to fit a power curve to observed learning of a keyboard with 10 keys, the model has no way of predicting the learning of a keyboard with 20 keys or judging what would happen to learning and visual search if one of the keys were moved around after learning, or its shape or colour were changed.

Cognitive models, in contrast, are grounded on psychological hypotheses on how agents make decisions. Often these models are implemented within a larger theoretical structure, an *architecture*, which are unified systems for integrating different components of thought and behaviour into one coherent cognition. Popular *cognitive architectures* include *EPIC* (Kieras and Meyer, 1997), *ACT-R* (Anderson, 2007), and *SOAR* (Newell, 1990). They can all implement modules or components that hypothesise, explain, and predict various cognitive abilities, such as memory, vision, motor control, and decision making. The models using these architectures are defined with if-then rules or *production rules*, which map from the states of the task environment and the internal states of the cognitive architecture into actions, that modify these internal states or manipulate the environment. The key idea in cognitive architectures is that they can incorporate multiple resources, such as motor and vision, to simulate task performance in arbitrary environments - as long as the interface between the environment and the architecture is defined, and correct productions have been modelled.

For the purposes of modelling learning of visual layout, both *ACT-R* (Nyamsuren and Taatgen, 2013; Salvucci, 2001) and *EPIC* (Kieras, 2011; Kieras and Hornof, 2014) have vision models. These can predict how visual features, such as colour, shape, and size, affect visual search patterns a the level of individual fixations. Although both architectures are suitable for modelling visual search of layouts, *EPIC* typically considers expert performance and does not simulate learning (Kieras and Meyer, 1997). Conversely, *ACT-R* has its roots in the modelling of learning (Anderson, 1976), which makes it suitable for modelling learning with graphical UIs (Fleetwood and Byrne, 2006; Jokinen et al., 2017). The model presented here is not directly implemented within the ACT-R architecture. The vision and memory models are derived from it, but with simplifications, that could be done due to the present model being used only for a specific task.

2.2. Modelling goals

With the model presented here, our aim is to cover phenomena in visual search and layout learning. As previously stated, the model should predict the following: (1) visual search of layouts, accounting for features of the layout elements; (2) learning of layouts; and (3) re-learning of layouts. This will be accomplished by developing a model of adaptive feature guidance, which integrates the models of vision and memory reviewed above.

The model aims to optimise its search behaviour, minimising search time given constraints and resources (Howes et al., 2009). EMMA (Salvucci, 2001) and PAAV (Nyamsuren and Taatgen, 2013) will be used to impose time constraints on the visual search of the model, while providing the model with vision; a model of LTM (Anderson et al., 1998) will provide the model with a resource for recalling location and feature information, again with time constraints; and utility learning (Anderson, 2007; Rescorla and Wagner, 1972; Veksler et al., 2014) will be used to simulate how the model learns optimal behaviour, given its resources and constraints.

3. Modelling adaptive feature guidance

We model visual search as **visual sampling problem**: the visual system must decide where to attend next to find a given target. To support this decision, the system must be assumed to have access to a representation of the visual elements (and features thereof) that make up the interface. Representation of the visual world by the model is incomplete on account of visual acuity limits, which we model by giving thresholds to various visual features. If an element is beyond a threshold from the model's current position of the eyes (the fixation), its visual properties are not considered in the bottom-up and top-down calculations. Elements with distinctive features are salient, and this guides attention towards them (bottom-up guidance). In addition, the model can be requested to try to match certain visual features (top-down guidance). Attention is then guided towards elements with features matching this top-down request. The bottom-up saliency and top-down match scores are combined, and the visual element scoring highest is attended next. An attention request is followed by a rapid eye movement (saccade) towards the target and a subsequent encoding, unless the target is close enough to the current fixation to be encoded without a saccade.

The model is adaptive in three ways. First, the model of associative long-term learning stores the locations and visual features of layout elements for future retrieval. The probability that the model will be able to retrieve the details about the target, along with the probability that the retrieval will be successful, depends on how often the model previously encountered the target and on how far in the past these encounters occurred. Repeated and recent exposure strengthens the associative connection and makes retrieving the location and visual features of an element more probable and faster. Second, the model uses short-term visual memory to inhibit revisitation of recently seen graphical elements. Contrasted with between-task associative memory, this memory spans only one task at a time, but has the benefit of being immediate and not having to build up gradually. Thirdly, the model tries to learn the optimal search policy, that is, it learns what types of feature requests have worked in the past search tasks. This is also a type of long-term memory, but instead of explicit associations, it implicitly models the fine-tuning of the model's search policy via reinforcement. The described flow of information from the memory resources to the controller, and of requests from the controller to vision, are depicted in Fig. 1.

We first describe the model's mechanism in detail below. Then, in the following section, we use the model to work a set of simple examples in order to demonstrate how the model works. Finally, we use the model to make predictions about real user interfaces, and use human data to investigate how well the predictions fit to observations.

3.1. Attention and eye movements

3.1.1. Feature guidance

The goal for the model is to find the target element by encoding visual elements of the environment. Encoding an element allows the model to decide whether it is the target or a distractor. Before the model can encode any elements, it needs to attend one. The feature guidance component holds a visual representation of the environment, and at the controller's request it resolves the request to deploy attention to one of the elements in it. The attended target is determined by the properties of the visual elements. The presence of these properties in the visual representation is based on their *eccentricity*, or angular distance from the fixation. A feature is visually represented if its angular size is larger than

$$ae^2 - be, \tag{1}$$

where e is the eccentricity of the element (in the same units as the angular size, i.e., in degrees) and a and b are free parameters that depend on the visual feature in question. Their settings, from the

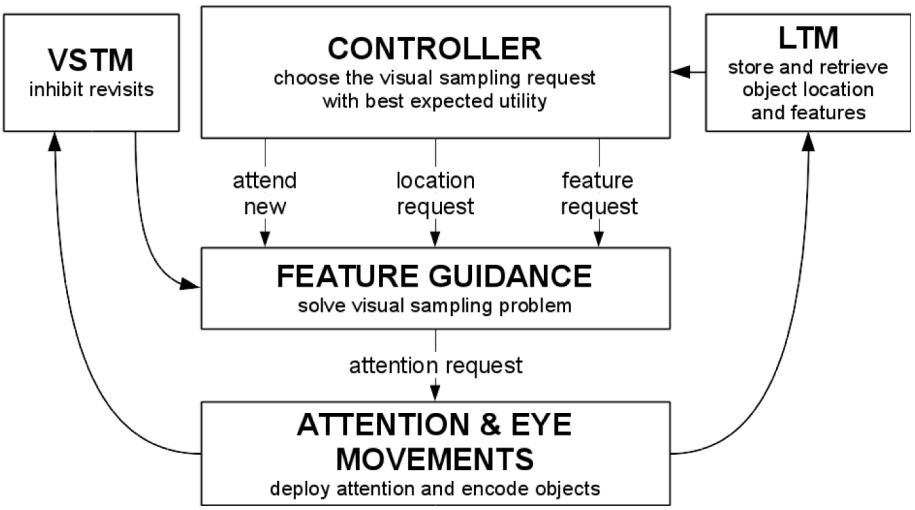


Fig. 1. The model is based on the principles of feature guidance and utility learning. On the basis of expected utility, the controller requests guided attention deployment from the eye-movement system. This directs attention to the most salient unattended visible element within the visually represented environment, and results in its encoding. If locational or feature information is accessible in the LTM, the controller, learning the utilities of its actions, can optionally also request these features to be considered in the attention deployment. Encoded elements are stored in VSTM, which inhibits revisits. Location and visual features of the elements are stored in LTM for future recall.

literature (Nyamsuren and Taatgen, 2013), are $a = 0.104$ and $b = 0.85$ for colour, 0.14 and 0.96 for shape, and 0.142 and 0.96 for size.

On the basis of the visual features represented, each element is given an activation as a weighted sum of bottom-up and top-down activations (Nyamsuren and Taatgen, 2013). Bottom-up activation is the saliency of an element, calculated as the dissimilarity of its features to all other elements of the environment, weighted by the square root of the linear distance between the elements:

$$BA_i = \sum_{j}^{elements} \sum_{k}^{features} \frac{dissim(v_{ik}, v_{jk})}{\sqrt{d_{ij}}}. \quad (2)$$

Two elements i and j are dissimilar, that is, $dissim(i, j) = 0$ for a feature if this feature is shared exactly between them (otherwise $dissim(i, j) = 1$). For instance, two green elements are similar, but a red element and a green element are dissimilar (on the colour feature). Hence, bottom-up activation of an element increases if it is close to elements that do not share its features. For instance, a unique red element among a large number of green elements gives the red element a large bottom-up activation, as it is dissimilar to all of the close elements. Conversely, the green elements have less bottom-up activation, because while they are dissimilar to the red element, they are similar to each other (on the colour feature).

If the controller does not include a set of features in the attention deployment request, attention is guided towards the element with highest bottom-up saliency (that is currently within the visual threshold of the model). However, the controller can optionally include a feature set to be matched in the attention deployment that results in a top-down guidance of attention.

Top-down activation entails the similarity of the feature set of the element to an optional controller-requested feature set (Nyamsuren and Taatgen, 2013):

$$TA_i = \sum_{j}^{features} sim(f_{ik}, f_j), \quad (3)$$

where similarity between the model-requested feature f_k and the element's feature f_j is 1 for a match, 0 for a mismatch, and 0.5 if the property f_{ik} is not present in the model's vision.

The total activation of an element is the sum of bottom-up and top-down activations, weighted by constants ($W_{BA} = 1.1$ for bottom-up and $W_{BA} = 0.45$ for top-down), plus noise from a logistic distribution with $SD = \sigma_{TA} = 0.376$ (Nyamsuren and Taatgen, 2013). An attention deployment request by the controller results in attending the element with the highest total activation. This is a simplification of the Guided Search Model, which models the guidance of visual search from bottom-up and top-down information (Wolfe, 2007).

3.1.2. Eye movements

After an attention deployment has been resolved, the model needs to attend and encode the element with the highest activation, as calculated above. The eye-movement component constrains the model by enforcing the encoding time as a function of the eccentricity of the element. It also provides a resource for moving the eyes closer to the target in order to minimise eccentricity. Aside from noise associated with eye movements, the fastest possible encoding time for any given element is guaranteed.

Our implementation follows the EMMA integrated model of eye movements and visual encoding for calculating eye movement and encoding time (Salvucci, 2001). What EMMA does not provide is the attention request, which comes from the model described in the previous section. The time to encode an element is

$$T_e = K \cdot [-\log(f)] \cdot e^{k \cdot \epsilon}, \quad (4)$$

where K and k are constants, and f is the frequency of the element, set to 0.1 if no frequency information is available, and in any case provided by the modeller or computed from the set of all possible elements that the model can encounter in its current task environment. ϵ is the eccentricity – measured as the distance of the target from the current eye fixation (in degrees). Because encoding time increases exponentially as a function of eccentricity, the visual system may initiate a saccade to get closer to the target. Saccade duration is

$$T_s = t_{prep} + t_{exec} + D \cdot t_{sacc}, \quad (5)$$

where t_{prep}, t_{exec}, and t_{sacc} are constants related to the human visual system and D is the distance to be covered by the saccade, in degrees. The landing point of the saccade is the target location with noise added from a normal distribution with a standard deviation of σ_V multiplied by the distance between the saccade starting point and the intended landing point (Salvucci, 2001). If the encoding time (Eq. (4)) is less than t_{prep}, then the target is encoded without the eyes moving from the current fixation location. If not, the saccade shifts the fixation location close to the target and the remaining encoding is conducted after the saccade. Where the feature guidance component of the model expresses thresholds for the visibility of features, the EMMA model expresses the threshold of *foveated area*. Only within this area can the model encode detailed information about the visual elements, such as text or small icons.

3.1.3. Visual short-term memory

VSTM holds a list of recently encoded elements, and inhibits the model from attending them. The parameter τ controls how long elements are held within the VSTM. Its value has often been set to $\tau = 4$ s (Jokinen et al., 2017; Kieras, 2011; Nyamsuren and Taatgen, 2013), but

for larger visual search tasks it has also been made practically infinite, that is, large enough that the search task is finished before the first elements start to decay (Kujala and Salvucci, 2015). Experiments with large sets of images have indeed suggested that the VSTM is not limited to a handful of items or decay times of a few seconds (Endress and Potter, 2014). Below, we note that long VSTM decay time works better for our model of layout learning, with the exception that an explicit τ value is necessary in simulating visual search of large layouts. Essentially this will inhibit revisiting of already encoded elements. While such revisits occur, they are rare and stochastic (Fleetwood and Byrne, 2006); therefore, we simplify the model by inhibiting return, unless the layout is very large. In reality, the human ability to seek a visual target among a large number of distractors is possible via a combination of bottom-up inhibition mechanisms and top-down search strategies (Fleetwood and Byrne, 2006), which we will not model here. However, below we will discuss how to empirically investigate and computationally implement such strategies.

3.2. Learning

The model has two long-term learning components for adapting to the task environment's and its own information-processing constraints.

3.2.1. LTM Update

Whenever the model finds a target, the LTM component enters the association of the target and its visual features and location in storage. The association strength is a function of the number and timestamps of storage entries: each time an entry is revisited, its activation gets stronger (Anderson et al., 1998). In contrast, the further in the past the entries are, the less they contribute to the activation, resulting in decay of memory. The activation of an element i is

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right), \tag{6}$$

where t_j is the time since the jth visitation of i and d is a decay parameter (Anderson et al., 1998).

In order to utilise information in the LTM to guide attention, the controller sends a retrieval request to the LTM. If $B_i > 0$, with added noise from the logistic distribution with a standard deviation of σ_M, and a base-level activation constant B_{bl}, the information associated with the current target is retrieved. In addition, source activation of the size of B_{sa} can be added to any B_i if entry i corresponds to a controller-requested feature (this is similar to the top-down guidance of attention). This can be used to spread activation to certain types of memory entries, such as those that are from the newly learned layout (vs the old layout). Time to retrieve information about an element i depends on its activation in the LTM and is given by

$$T_i = Fe^{-fB_i}, \tag{7}$$

where F and f are individual scaling constants (Anderson et al., 1998). If recall is successful, the controller can use the recalled feature information to request attention deployment, using the top-down feature guidance described above. Alternatively, the model can request attention deployment to the recalled location of the element.

3.2.2. Utility learning

In order to decide whether to proceed with the search by using VSTM inhibition or a retrieved feature or location information, the controller uses utility learning. When a target has been found, the model rewards all actions used to find it, discounted by time. The utility u_i of an action is updated via the *delta learning rule* (Anderson, 2007; Rescorla and Wagner, 1972; Veksler et al., 2014):

$$\Delta u_i(t) = \alpha[1 - R_i(t) - u_i(t-1)], \tag{8}$$

where α is a parameter to be estimated, $R_i(t)$ is a temporally discounted reward (set to a unit value of 1), and $u_i(t-1)$ is the utility at the previous step. The action with the highest utility, after addition of noise from a logistic distribution with standard deviation σ_U, will be taken. The utilities dynamically adjust to the changes in the environment. For instance, if the layout is changed by moving the elements around, actions that utilise retrieved location information might receive lower utility values, because they do not work well anymore.

The Eq. (8) can be considered to implement *temporal difference reinforcement learning* (Sutton and Barto, 1998). However, it should be noted that our model does not employ any traditional learning algorithm within the reinforcement learning domain, such as Q-learning or SARSA. Instead, we update the utility values of each action by updating its utility using the delta learning rule on time discounted reward at the end of each search task. This follows the usual practice of reinforcement learning implemented in ACT-R models, which has the benefit of matching learning observed with human participants (Anderson, 2007). We summarise all parameters of the model and their values in Table 1.

Table 1
Parameters of the model and their descriptions. Literature-based values are cited.

Parameter	Component	Definition	Value	Ref.
k	VISION	scaling factor for encoding speed	0.4	Salvucci (2001)
K	VISION	scaling factor for encoding speed	0.006	Salvucci (2001)
t_{prep}	VISION	visual pre-processing time	0.135	Salvucci (2001)
t_{exec}	VISION	saccade execution time (baseline)	0.070	Salvucci (2001)
t_{sacc}	VISION	scaling factor for saccade duration	0.002	Salvucci (2001)
σ_V	VISION	saccade landing point noise	0.1	Salvucci (2001)
a, b	VISION	thresholds for feature vision	*see (1)*	Nyamsuren and Taatgen (2013)
W_{ba}	VISION	Weight for bottom-up activation	1.1	Nyamsuren and Taatgen (2013)
W_{ta}	VISION	Weight for top-down activation	0.45	Nyamsuren and Taatgen (2013)
σ_{TA}	VISION	Noise for total visual activation	0.376	Nyamsuren and Taatgen (2013)
τ	VSTM	decay (forgetting)	45	–
d	LTM	decay (forgetting)	0.5	Anderson et al. (1998)
F	LTM	scaling factor for overall retrieval times	1.06	Jokinen et al. (2017)
f	LTM	scaling the effect of activation on retrieval time	1.53	Jokinen et al. (2017)
B_{bl}	LTM	base-level activation	6.0	–
B_{sa}	LTM	source activation for top-down control of retrieval	3.0	–
σ_M	LTM	noise for retrieval activation	0.6	Jokinen et al. (2017)
α	CONTROL	scaling factor for utility learning	0.1	–
σ_U	CONTROL	noise in utility calculations	0.3	–

4. Model walkthrough

4.1. Feature guidance

4.1.1. Bottom-up activation

In order to demonstrate the basic guidance principles of the model, we walk through its simulation of a simple visual task. Fig. 2 shows a layout with four elements. The fixation of the model is at the left border of the layout (marked X in the figure). This fixation point is purposefully selected to illustrate the visual thresholds of the model. To the right of it, there are rectangles A–C aligned in a single column, and further to the right of them is a single rectangle D. One of the three closer rectangles is green (C), and all other rectangles are blue, including the one that is farthest on the right. The distance of the fixation (X) from the centre of D is set to 10 cm, and the distance of the model from the screen is set to 60 cm such that the visual angle between X and D is approximately 9.5°. The angular size of each rectangle is 1.5°.

Assuming that the model has no top-down feature requests available, that is, it does not prefer any features, its attention deployment is based on bottom-up activation of the elements in a winner-take all fashion: highest activation dictates the element that is attended to next (Itti and Koch, 2000; Nyamsuren and Taatgen, 2013). Before calculating these activations, Eq. (1) is used to update which of the features are visible to the model, given its current fixation point (Kieras and Hornof, 2014). As an example, we calculate here the features of the farthest rectangle (D) that are available to the visual system of the model. The requires the visual angle between the eyes and the element, the angular size of the element, and the feature-specific parameters a and b. For the colour feature, Eq. (1) gives $0.104 \cdot 9.5^2 - 0.85 \cdot 9.5 = 1.311 < 1.5$, which means that the visual system of the model is able to perceive the colour of the element. However, shape is beyond the threshold, $0.14 \cdot 9.5^2 - 0.96 \cdot 9.5 = 3.515 > 1.5$. Similarly, the relative size of D is too far to be perceived by the visual system. For the closer rectangles A–C, the same equations produce perceivable colour, shape, and size. Features within their corresponding thresholds are added to the visual representation of the model.

After the model's visual representation of the layout has been updated, the bottom-up activation of each element is calculated by comparing its visible features to the visible features of all elements using Eq. (2). Because element C has a dissimilar colour to the other elements, its activation is highest at 0.71. Element A has the lowest activation, 0.51, because it is similar to B, and although it is dissimilar to C, its distance to C is larger than B's distance to C, whereas B gets a bottom-up activation of 0.59. The activation of D is 0.62, which is larger than for A and B, because its shape and size are not visible, which contributes toward D being dissimilar to the other elements (features not present contribute towards dissimilarity). It should be noted that although D is not differently shaped and coloured than A and B, this does not matter here because the model does not currently see these features. Were the model to refixate in the middle of the layout so that all features of all four elements are visible to it, D would receive the lowest amount of activation, with C again having the highest, then B, and finally A. With no top-down requests, the weighted sum of bottom-up and top-down activation of an element is simply W_{BA} times the activation calculated above, with added noise from a logistic distribution. As the noise is the same for all elements, the most probable target of the model's attention deployment is C. This is because C is the most salient visible element, having a dissimilar colour to all other close elements.

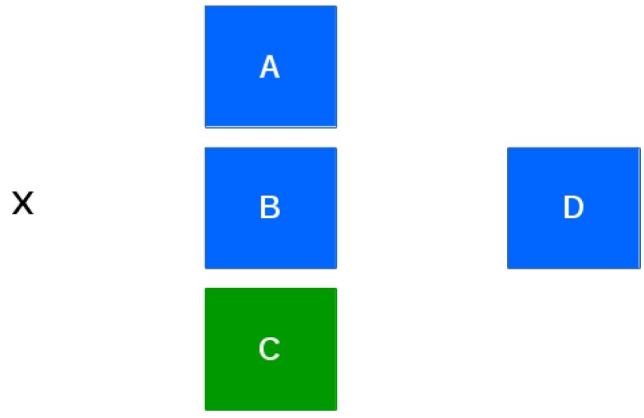


Fig. 2. Example layout for the model. The letter X shows where eyes are fixated. Element C has the largest bottom-up activation due to its distinct colour. The model can encode elements A–C without a saccade, but D requires a saccadic movement to bring fixation (X) closer to it.

After the target for attention deployment has been selected, the model encodes the target element, which is 3.35deg away from the current fixation. The encoding time in seconds is given by Eq. (4): $0.006 \cdot [-\log(0.1)] \cdot e^{0.4 \cdot 3.35} = 0.05$. Because this is smaller than the visual pre-processing constant t_{prep}, the model does not need to execute a saccade towards the element, but instead is able to encode its contents from the current fixation and decide whether it is the visual element that is being searched. This demonstrates the relative costliness of eye movements compared to encoding elements without a refixation. For comparison, if the model were to encode the element D, the encoding time would be 0.23s. due to the fact that in order to quickly encode the target, the model needs to prepare ($t_{prep} = 0.135$) and execute a saccade close to it ($t_{exec} = 0.07$, plus each travelled degree has an additional cost of $t_{sacc} = 0.002$). In other words, it is possible to encode the element C while fixating on X, but a saccadic movement is required for encoding element D. After the encoding of C, due to inhibition of return it would not be considered again as a valid target for a duration (governed by τ), whilst the search progresses. It would, however, still be used normally in the bottom-up calculation of other yet unvisited targets.

4.1.2. Top-down activation

The previous example assumed that the model has no prior knowledge of the layout, which forces it to treat all elements as equally probable targets. However, if the model is given additional cues, such as information that the element it is looking for is a blue rectangle, it can use this information for its attention deployment request. This adds top-down activation to all elements sharing these features - assuming of course that these features are visible to the model. Top down activation of each element visible to the model is calculated from Eq. (3). For elements A and B in Fig. 2, both colour (*blue*) and shape (*rectangle*) are a match, and thus both get $1 + 1 = 2$ as their top-down activation. Element C has a feature match for shape but not colour, and thus gets a top-down activation of 1. Element D matches for colour, but this feature is below the visibility threshold, and thus missing in the visual representation of the model. A missing feature contributes 0.5 to the top-down activation of element D, giving it a total top-down activation of 1.5. In order to decide where the attention is deployed, total activation of each element is calculated by multiplying bottom-up activation by W_{ba} and top-down activation by W_{TA} and summing them. This gives the following activations: A = 1.46, B = 1.55, C = 1.23, D = 1.36. Assuming no noise, the model deploys attention to the element with the highest total activation, which in this case is B. This is because B matches both feature requests and, in comparison to A, is more salient due to proximity to the differently coloured C.

4.1.3. Model input

The model described here deals with the visual world *symbolically*. This means that each visual element is given to it as an object, which has slots for location, size, and features. In other words, the model itself does not deal with the problem of extracting visual elements from a scene, such as a screenshot of a layout. Instead, the modeller needs to specify – automatically or manually – the visual elements along their specifications (this is similar to how the cognitive architecture ACT-R

deals with its visual world, see Fleetwood and Byrne, 2006). All features of the elements are qualitative in that the model makes absolute comparisons between them. For example, colours *green* and *light green* are as distant from each other as from *blue*. It is up to the modeller to define the features such that the model's visual world corresponds to the actual layout. Further, as will be shown below, the model does not learn to use icons or other semantic visual aids to guide its search.

4.1.4. Feature search

The model searches for a given target by requesting attention deployment, which selects the visual element with the highest total activation as calculated above. This results in an attention deployment request and the subsequent encoding of the element, potentially via a saccade if the target is sufficiently far from the current eye location. Although the model does have feature information concerning the element, we assume here that each potential visual target contains some detailed information, such as text or an icon, which can only be encoded within the foveated area. Based on the encoding of this detailed information, the model decides if the encoded element matches the target it is looking for. Assuming it does not, the search continues: the model flags the currently attended element into its VSTM to prevent it from being attended to again in the near future, and calculates a new highest active visual element.

If the model has no feature information to guide its search, average search time increases as the number of elements increase (assuming that the target is not especially salient, i.e., it does not have bottom-up advantage). However, if the model has information about the features of its target, it can use this to shorten the visual search. Fig. 3a illustrates one of the four example search conditions, modelled after a classic search task (Treisman and Gelade, 1980). The target (marked with *) is surrounded by a number of other elements. The conditions differ dependent on the set size (small = 8 or large = 16 elements) and whether half of the elements share the colour of the target (conjunction) or the target is the only element of its colour (disjunction). Based on the description of the model, it is easy to predict that in the conjunctive task, set size impacts search times, whereas in the disjunctive task, adding more elements should not increase average search times considerably. The results of model simulations are shown in Fig. 3b.

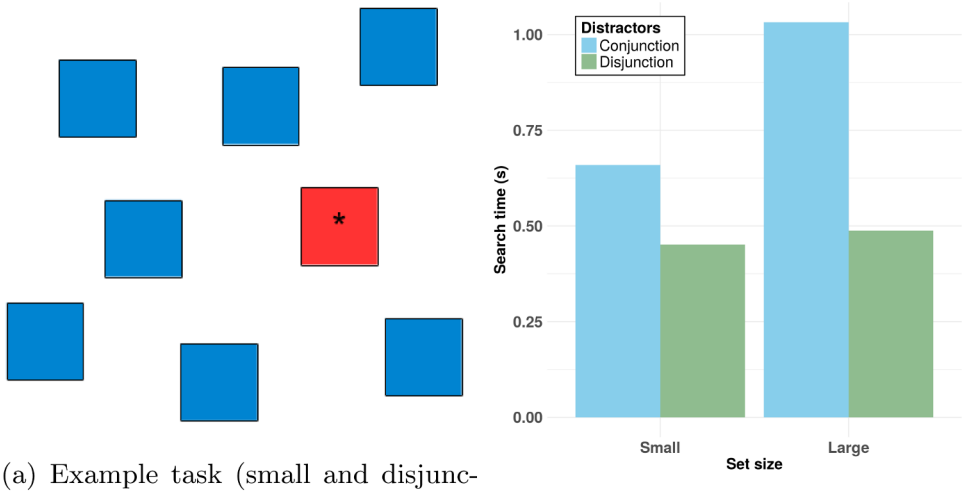
4.2. Layout learning

The feature guidance part of the model treats every visual search task in isolation. The memory component, in contrast, stores information between tasks and provides this information to be used top-down in individual search tasks. In the above example, Fig. 3, the attention was deployed using the top-down information that the target is red. Where did this information come from? As in the original experiments demonstrating the effect with people (Treisman and Gelade, 1980), the information was explicitly given to the model before the task began. When people use graphical interfaces, instead of being provided the top-down information, they *learn* the relevant information so as to make their upcoming use more efficient. Although our model is not an instance of the original Feature Integration Theory by Treisman and Gelade (1980), it should still be able to replicate these results. Please see Wolfe (2007) and Nyamsuren and Taatgen (2013) for discussion on the differences between that theory and the guided search theory, that is adapted in our model.

After the model has found the target it is currently searching, it stores the features of the element as well as its location into its associative memory for future retrieval. Whenever it starts a new search task, it sends a retrieval request to this associative storage. Because the retrieval always takes some time, the model starts the normal bottom-up driven visual search, as described above. If the retrieval is successful, the model adds the retrieved feature information to the top-down feature requests, also as described above. In addition to this, it can directly attend to the element that is close to the recalled location.

An example simulation of the model's feature and position learning is shown in Fig. 4. The model starts its search tasks as a complete novice. It is randomly given a target to search for, and the search time is recorded. In the beginning, as a novice, the model takes about 1.4 s to find non-salient elements, such as the *Messenger* and *settings* elements (shown in the bargraphs in Fig. 4). Conversely, the green *Excel* element is salient due to its distinct colour, so search times for it are already fast in the novice phase, that is, even without any top-down guidance. This is the same result as demonstrated above in Fig. 2, except now the search times are longer due to a larger number of searchable elements. After 10 simulated minutes of constantly searching for random elements on the layout, the model can be considered to hold expert knowledge of it. It now takes less than 0.8 s to find any of the three elements used in the example. This is because it remembers their positions well, and can use this positional knowledge to guide attention toward the right area. Due to the noise in the various resources of the model, such as attention, eye-movements, and memory, the most salient element remains on average the fastest to find.

After the model has reached expert level, two of the three target elements are moved around. The model still first attends to the old location, because this positional knowledge has proven to be an efficient way to guide attention. In other words, the utility of the location request is high, and thus so is the probability that it is used because this request has usually resulted in fast search times. After the model



(a) Example task (small and disjunctive)

(b) Results of feature search

Fig. 3. The model searched for the red target (*) in four conditions (size × conjunction). The number of elements was 8 or 16. In conjunctive layouts, half of the elements were red (the same as the target), and in disjunctive, only the target element was red. The results show that increasing set size does not have a notable effect on search times in the disjunctive condition, but in the conjunctive condition it increases the search times. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

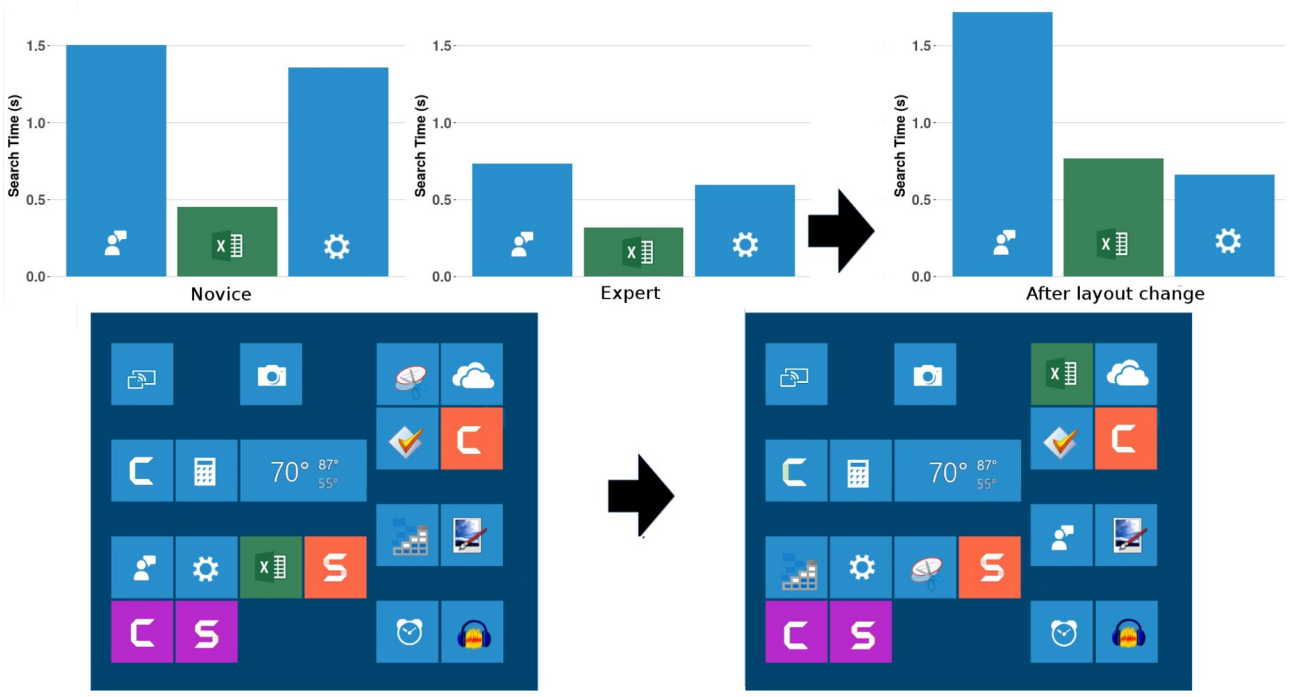


Fig. 4. The visual search model predicts visual search times for new and changed layouts. For a novice user without any prior exposure to the layout, the model predicts that of the three elements chosen for this comparison, the salient green element is the fastest to find. After learning the locations of the elements, the expert model finds all fairly quickly. At this point, one blue element and the green element change place. Search times for the moved blue element are longer than for the green element, because the model remembers the distinctive features of the latter. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

observes that the target is not at the location it expected, it gives negative reinforcement to the location-based attention guidance. This results in a shift in search strategy, and the model starts to use the recalled feature information, such as colour, to guide attention in a top-down fashion. As a result, the immediate search times for the moved elements increase, especially for *Messenger*. For this element, the feature-based top-down guidance of the model does not result in a faster visual search since the element shares its features with many other elements.

In comparison to *Messenger*, the green *Excel* is still relatively easy to find even after the layout change. This is because the top-down feature guidance for green colour matches only this element, which results in it receiving higher total activation than other elements. Using reinforcement learning, the model shifts its future strategy so that it relies less on the unreliable positional knowledge of the moved elements. For *Excel*, it learns to rely on feature knowledge. After some time, the model has learnt the new locations of the moved elements, and accordingly it starts to shift back to the more exact position-based search. It can do this, because the strategy it follows is updated constantly, via the delta learning rule (Eq. (8)), based on how quickly the use of different information sources result in finding the target. The *settings* element serves as a control. The search time for it does not change, except for small variations due to changes in the layout, which results in subtle changes in its bottom-up activation. Here, as its green neighbour is replaced with a blue element, its bottom-up saliency decreases slightly, which results in slightly longer average search times.

5. Validation with human data

To test the model under realistic conditions with multiple different layouts, and to fit the free parameters of the model, we conducted an experiment. As the model predicts visual search on layouts, the experiment was designed to have human participants visually search for cued elements in realistic graphical UIs. The model also predicts learning of locations and features of visual elements as the function of repeated visual search, so the participants repeated the search tasks multiple times. Finally, as the model predicts how a partial layout change impacts visual search and relearning, the participants were also given tasks with modified layouts after they had first learned the original layout. The main dependent variables produced in the experiment are search time and eye movements over the layout. Independent variables are layout, target with its features and locations, and the number of repetitions. To confirm the representativeness of the model, that is, its applicability to different interface types, we chose layouts from separate domains: a consumer interface (ticket-vending machine), a website (*New York Times* front page), and a computer OS interface (Windows 10 menu); see Fig. 5.

Most of the parameters of the model were adapted straight from existing literature. However, certain parameters, especially those related to the activation calculations in the LTM component of the model, are context-dependent and thus were fitted to the human data. Hence the data from the experiment were first used to acquire the parameters which provided the best fit. After that, quantitative and qualitative comparisons of the model predictions and human observations were used to assess the model's validity.

5.1. Method

In total, we recruited $N = 20$ university students, aged between 19 and 33 years, $M = 24$, $SD = 4.56$. All were regular computer users and familiar with using the internet. The experiment applied a within-subjects design wherein each participant performed a task of visually searching for visual elements in graphical UIs. The tasks were modelled after visual search experiments in which the participants were asked to visually locate the cued target as quickly as possible and then to press a reaction time (RT) button in front of them to indicate that the target had been found (Jokinen et al., 2017; Sears et al., 2001).

Fig. 5. The three layouts used in the experiment. Left: a ticket wending machine (BP); centre: a news site (NYT); and right: a desktop OS menu (WIN10). For larger images, see the Appendix (Figs. 13–18).

5.1.1. Stimuli

Screenshots of three distinct layouts served as stimuli: a Budapest Transport Authority (BP) ticket-vending machine, Windows 10 desktop screen, and *New York Times* front page (see Fig. 5, and the Appendix Figs. 13–18 for larger images). In addition to the using of each layout as is, we modified them slightly by moving some of the elements around. Each participant learned each layout by looking for cued targets in it for 10 min. After this, the layout was changed to the modified layout and the participants retrained themselves with it for another 10 min. The order of the layouts was Latin-square counterbalanced, but the within-layout order was always original first, modified second.

5.1.2. Procedure

At the beginning of the experiment, participants filled in the consent form. Then, the experimenter explained the procedure of the experiment, which consisted of a visual search task and a motor calibration task. The participants were allowed to stop the experiment at any time if they wished to, but all participants finished the whole experiment. In the visual search task, the participants were shown a screenshot of one of the three UIs, with a text cue on top of the layout. The cue was always given as non-formatted text that matched the text content of one of the visual objects in the stimulus image (possibly shortened slightly to fit the cue line). The participants read the cue and then located the corresponding target with their eyes, pressing the RT button as soon as they had found the target. After pressing the RT button, the screen was blanked for a short period (less than a second), and then a new cue with the same layout was shown. There were 10 individual targets for each layout, picked so that there was variability in the locations of the elements as well as in the visual features, such as the size and colour. The participants conducted search tasks for 10 min, after which the layout was changed to its modified version, followed by a new block of 10 min of searching. For each layout, four of the targets were moved in the layout modification.

In addition to the visual search task, we had the participants perform a motor-calibration task at the start and the end of the experiment. They were shown a large circle on the monitor for a random time in the range of 5–10 s. The task, performed five times, was to press the RT button as soon as the circle disappeared. The resulting motor-calibration data were used to adjust the search times recorded in the visual search experiment: each visual search time was reduced by the mean RT from the two motor-calibration tasks. The purpose of this task was to control for the variance in individual motor-movement time, as the visual search model does not simulate motor movement. The total experiment time for one participant was about 70 min, including 5 rests of 2–3 min between the layouts (but not between the layout change from the original to the modified one).

5.1.3. Apparatus

Eye movements of the participants were recorded during all visual search tasks via a Tobii X120 eye tracker with a sampling rate of 120 Hz. The resolution was set to 1680 × 1050 pixels (17-inch LCD screen), and the participants were seated approximated 60 cm from the screen. The participants were told not to cheat by pressing the RT button before actually visually locating the target element, and that eye-tracking equipment would be used to ascertain whether they had. No signs of fraudulent behaviour were evident in the data. We calculated the distance of the final fixation from the actual location of the target. The fixation does not, of course, need to be precisely on top of the target for the target to be encoded, but for all participants, the mean distance was within less than one standard deviation of the overall mean distance, aggregated over all participants.

5.2. Model calibration

Most of the model parameters, listed in Table 1 were taken from literature. This was done to ensure the persuasiveness of the fit indices. The remaining parameters were fitted via a grid search into the whole experimental data (all layouts, both versions). We ran 50 model instances per parameter combination. The parameter set with the best (smallest) *RMSE* was selected. For calculating *RMSE*, we averaged RT into 20 trial bins per layout (10 bins per layout version). We used the binning in order to have an aggregate measure of search times, because individual trials had a lot of variance in RT, depending, for example, on which cues were randomly assigned to the participants in that trial. The goal of the parameter fitting was to find a set of parameters that would result in small *RMSE*s across the three layouts.

5.3. Results

In total, there were 24,514 trials by the 20 participants. Of these, we removed 21 (0.008%) due to these being instances of extensive outliers (a trial taking over 80 s to complete). In average, the participants performed, per layout version, 240 trials with BP, 144 with NYT, and 238 with WIN10. In order to test the learning and relearning of the layouts, we used a mixed linear regression, which allows us to add the participant and the trial cue as random intercepts into the model (Judd et al., 2012). The model's dependent variable was RT, and trial, layout, layout version, and interactions between these were independent effects. The participant was added as a random intercept effect. The model is displayed in Table 2 (*t*-tests use Satterthwaite approximations to degrees of freedom). The NYT layout significantly increased average search times, but BP and WIN10 search times were, on average, similar. Each trial lowered the estimated RT statistically significantly, suggesting learning during the experiments. The impact of layout change was also as intended, increasing average search times, although the participants were then able to relearn the layout (demonstrating even faster learning than with the first layout).

The best-fitting simulation model parameters are displayed in boldface in Table 1 (p. 16). Because the layouts differ in their search times, we consider model fit separately for the layouts (but note that the

Table 2
Fixed effects in the linear mixed model predicting RT (s). *Intercept* is the first version of BP layout.

Effect	Estimate	*t*
Intercept	2.9	12.8*
NYT	1.0	3.7*
WIN10	−0.1	−0.5
trial	−0.01	−27.3*
version	−0.4	−9.5*

Note. *$p < .001$.

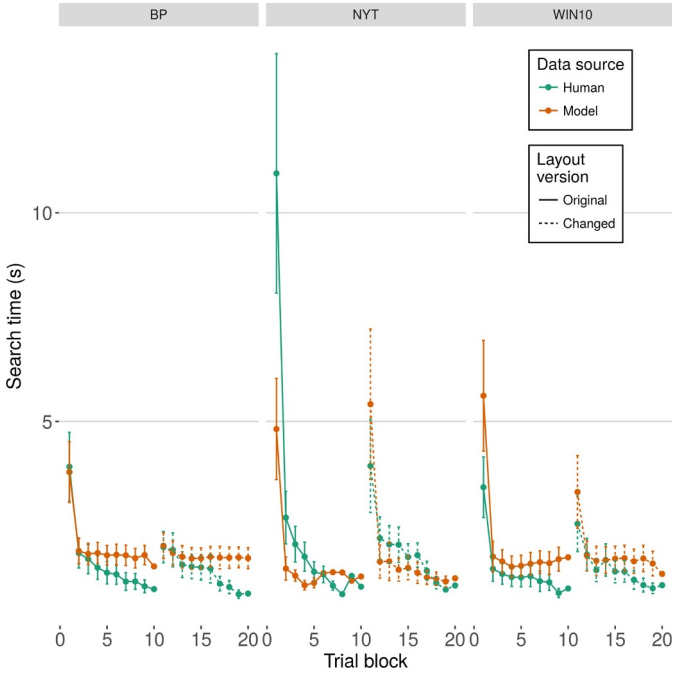


Fig. 6. Human and model search times in the three layouts, binned into 20 trial bins.

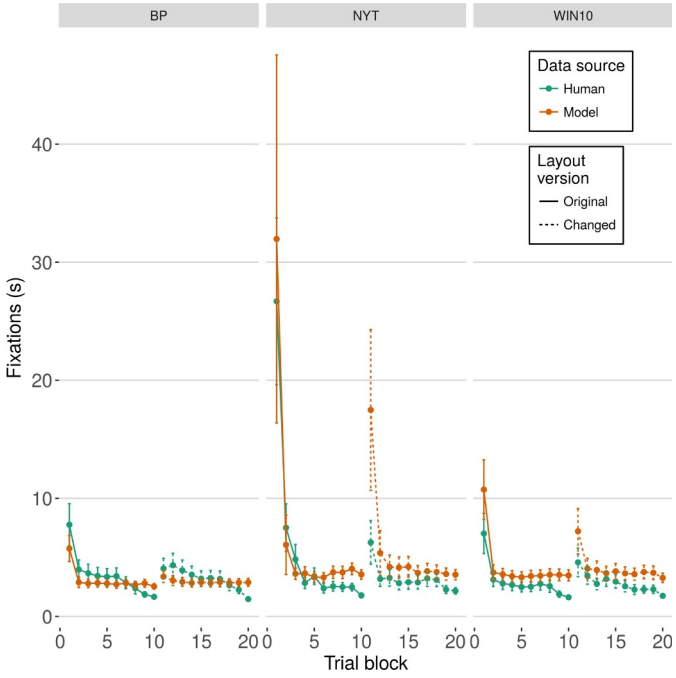


Fig. 7. Observed and predicted fixation counts for the three layouts in 20 trial bins. The error bars denote *SE* with $N = 20$.

Table 3
Model's linear fit and prediction error between the layouts for search times and fixations counts.

Layout	Search time		Fixations	
	R^2	*RMSE*	R^2	*RMSE*
BP original	0.94	0.45	0.84	0.9
BP modified	0.48	0.51	0.26	0.8
WIN10 original	0.95	0.73	0.92	1.6
WIN10 modified	0.78	0.42	0.74	1.4
NYT original	0.96	1.98	0.98	2.0
NYT modified	0.81	0.62	0.92	3.7

parameter fitting was conducted for all layouts together). Fig. 6 shows trial binned mean RTs for human and model data, and Fig. 7 shows the same statistics for fixation count. The raw trial-by-trial average Figure 7 RTs for human and model data are displayed in the Appendix (Fig 19). Using the global parameters that were fit for data from all layouts but calculating model fit for only a given layout (i.e., layout versions separately), we obtained fit indices reported in Table 3.

The differences in R^2s and *RMSE*s for modified BP can be explained by high overall performance: the data contain almost no variance aside from random variance, so, while the model *RMSE* is excellent, its R^2 does not reflect the model's successful prediction. Further, the human participants keep on improving with the layout, whereas the model's performance improvement flattens more quickly. The probable reason for this is that the human participants learn to recognise the relevant targets without having to read the whole text of the target (most targets in the BP layout had multiple words in them). Conversely, the model is always forced to read all text in the visual target before decision. In future work, the model could be augmented to learn to make the decision without having to read all of the text.

For WIN10, initial model search times are higher than for human participants. A closer inspection of eye tracking, seen in Fig. 8, demonstrates one reason for this. The human participant disregards the small visual items in the left and focuses on the large icons, whereas the model does not make this discrimination. It might be that in the beginning, without any previous experience with the layout, the human participants expect to find the targets in the main layout area, and not in the sidebar, based on their experience with layouts in general (there were targets both within the smaller and larger items, but most of them were among the large icons). Of course, the model does not have this kind of top-down guidance. Another probable reason for the higher initial search times for the model is that the human participants were able to visually chunk the layout, which consisted of a fairly large number of elements but was well structured. As the model has no such ability, its initial search times were higher. Nevertheless, the model was able to predict the quick learning of the layout and the impact of the layout change with subsequent relearning. For NYT, initial times were clearly longer than for the other two layouts; hence, the larger *RMSE*s are expected. Looking at the error bars of the first trial bin of NYT in Fig. 6, the error bars are very large for both the model and human search times.

An item-level analysis reveals that the feature-guidance assumption of the model predicts successfully the effect of visual features on search time. Fig. 9 reports observed and predicted average search times and variance for two different elements for the three layouts. More salient targets are more easily found in the novice phase, but after learning saliency does not matter. After a layout change, moved objects with salient features are easier to find. The model predictions fit the observations well except for the first learning phase of NYT. As already discussed above, our model does not capture the problems that the human participants had on their first trials on a layout with high information density. Cue-based R^2 and *RMSE* values are listed in the

(a) Human participant, novice search. (b) Human participant, learned search.

(c) Model simulation, novice search. (d) Model simulation, learned search.

Fig. 8. Scanpath comparison from human participant and model. Target was *Audacity*. Subfigures (a) and (c) show human and model predictions for the first time searching for the target. Subfigures (b) and (d) show the subsequent, learned search behaviour.

Appendix (Table 4). They generally show high fit, although at this granularity level, there is noise in the observations (a given cue at a given trial time bin might only have one or two participants' results due to cue randomisation). For some, such as the salient *Excel* icon in WIN10, the small R^2 but also good *RMSE* are due to both human and model learning to find the icon quickly, after which there is little variance to explain. This interpretation is also supported by the item-level analysis of *Excel* in Fig. 9.

The model was successful in predicting fixation count similarly to the search times, which is not surprising as the task involved only visual search. The R^2*s* and *RMSEs* for the predicted fixations are reported in Table 3. Again, the low fits of the NYT and modified BP layouts are explainable. For NYT, the initial search behaviour contained a lot of randomness, and our model does not capture the initial search behaviour of large layouts on a fixation level. The modified BP layout did not show any learning, so the curve was flat and noise was the primary source of variance.

Similarly to a study utilising the same eye-movement model (Jokinen et al., 2017), our work generally underestimates the number of fixations. The reason for this is probably that our model strictly refixates only if this takes less time than encoding without refixation. In comparison, human participants tend to make more small fixations around the given target while encoding it. In order to illustrate the eye movement behaviour of the model, we selected demonstrative examples, shown in Fig. 8. It compares scanpaths from a human participant and a model simulation in novice and learned trials. Initially, both the human participant and the model have to do visual search across the layout, which results in long scanpaths with many fixations. Here, the human scanpath is more structured. This is probably due to the reasons discussed above: the human participant visually chunks the search area for a more structured search. After the participant and the model have learned the position of the target, search is fast.

5.4. Modelling keyboard layout learning

We additionally tested the model by revisiting learning of keyboard layouts. As keyboard layouts are an instance of graphical layout, we expected that the model would perform similarly here to the keyboard-focused model (Jokinen et al., 2017). There are some differences between how attention is deployed in these models. The earlier, keyboard-focused model did not employ the notion of bottom-up and top-down activations, but instead preferred keys that were close to the current fixation. The model presented in this paper prefers visual elements which are most salient and optionally bear similarity to the top-down feature request. We did not fit any new parameters for this comparison; we used them as they were fitted above (see Table 1).

With the simulation time used in Fig. 3 of Jokinen et al. (2017), the model was trained for 2.5 simulated hours with a normal Qwerty keyboard layout in which all keys share exactly the same visual features. Our model predicts the average search times of 1.2 s in the first, novice phase, which corresponds to the 1.2 s reported (Jokinen et al., 2017). Likewise, after 2.5 simulated hours of learning, our model predicts the expected 0.7 s search times. To test the impact of layout change, we trained the model first for a Qwerty layout for 50 simulated hours, then switched to the Sath (Dunlop and Levine, 2012) layout. As in (Jokinen et al., 2017), Fig. 6, our model predicts that the keyboard change immediately increases average search times to 0.8 s. The additional benefit of our model to the existing keyboard layout learning model is in the feature guidance of attention. With our model it is possible to change the size, colour, and shape of the keys and investigate the effect of this on layout learning.

5.5. Sensitivity to parameter choices

In order to test the model's sensitivity to changes in the parameter values, we chose literature-based minimum and maximum values for each fitted parameter, and investigated how the model behaviour changes in these extrema. For each parameter, other parameters were set to their fitted values as in Table 1, and the model was run for the three layouts as above, once for the minimum and once for the maximum parameter value (listed in the Appendix, Table 5).

- **Retrieval time scaling *F***. Setting the parameter to the minimum of 0.1 has a slight negative impact on the search times at the beginning of the learning phase of the model. Raising the parameter value up to 3.0 does not discernibly change the behaviour of the model when

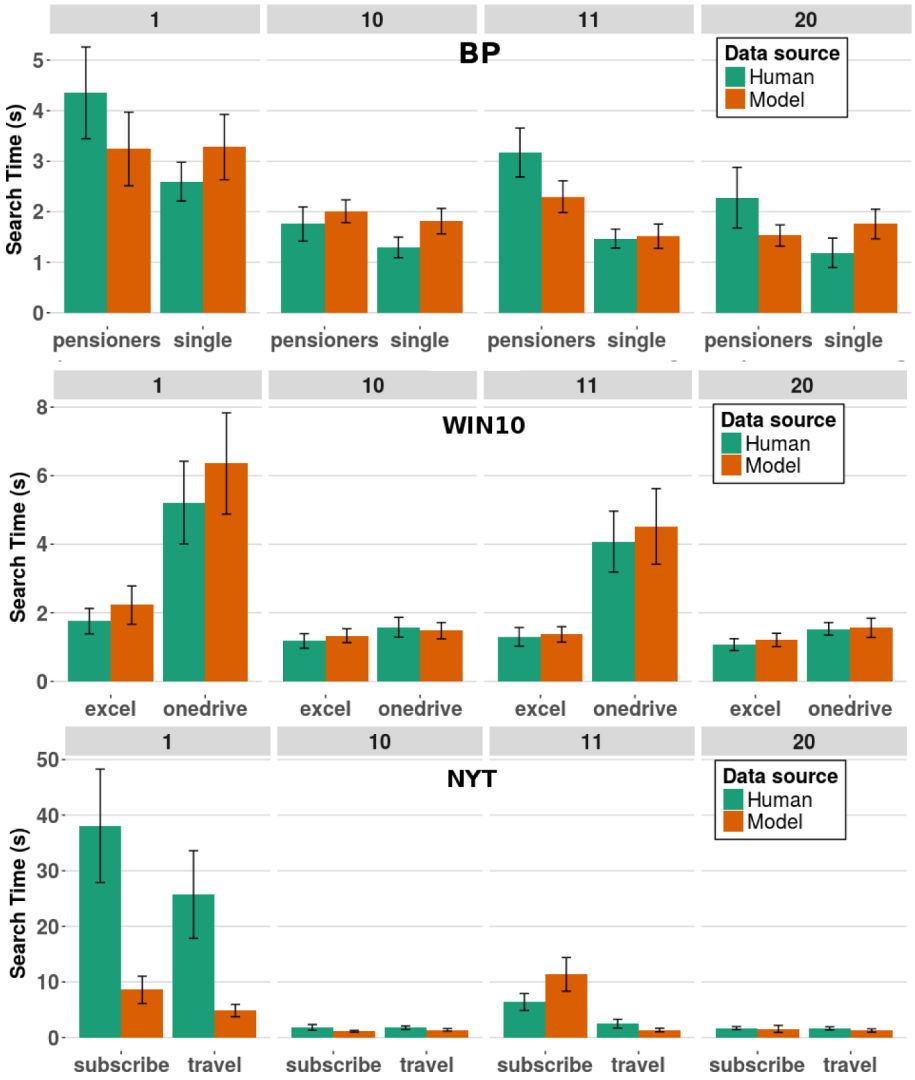


Fig. 9. Human and model search times in the initial and final phases for two elements in the three layouts layouts. In the beginning of layout learning (1 min), more salient targets (e.g., 'excel'), are faster to find than non-salient targets ('onedrive'). Right before the layout change (10), all elements can be found quickly. After layout change (11), moved elements are slower to be found ('pensioners', 'onedrive' and 'subscribe'). After relearning (20), the search for everything is occurs quickly again. Error bars are standard errors with $N = 20$.

compared to the literature-based value that was used above.

- **Base-level activation B_{bl}.** With the minimum value set to 0.0, the model, as expected, has difficulties in learning the layouts. This is because the memory entry decays so fast that the model is almost never able to recall the properties of the elements, and hence these memory entries are not being reinforced. With the maximum value of 10 the behaviour of the model is similar to when it is 6.0.
- **Source activation B_{sa}.** Setting the parameter to 0.00 was visible in poorer model performance in the relearning phase of the experiment. Source activation was used in the relearning to emphasise the top-down knowledge that the interface had changed to help the model learn the new interface in the face of strong activations from the old interface.
- **Visual short-term memory decay τ.** Disabling VSTM completely by setting the parameter to 0 leads to the model not being able to proceed in search tasks of WIN10 and NYT, while the impact on BP is very small. This is understandable, because without being inhibited from revisiting already seen graphical elements, the model has practically no hope of finding the targets except by accident. Setting the value beyond the calibrated value does not change the behaviour of the model discernibly. Of the fitted parameters, τ, is considerably larger here than is often the case in similar models (Kieras and Hornof, 2014; Nyamsuren and Taatgen, 2013), although some studies suggest that VSTM capacity is greater than suspected (Endress and Potter, 2014). Especially for NYT, τ needs a specific minimum value in order to simulate the effect of layouts with a lot of content. As discussed above, the inhibition of the revisit of encoded elements probably is explainable not by one parameter but, instead, by search strategies and multiple cognitive resources. Here, we simplify these into a single parameter τ, which is enough for a model of UI layout learning.

For the other parameters, changing their values between the theoretical extrema did not have a qualitatively discernible impact on the performance of the model. It is important to note that although fine-tuning the parameters changes the performance of the model slightly and helps to fit the model, the model is largely inflexible in the face of these parameter manipulations. This inflexibility is a good result, because it means that the architecture of the model, not its parameters, is the main constraining element in the model. This gives credibility to the model architecture (Roberts and Pashler, 2000).

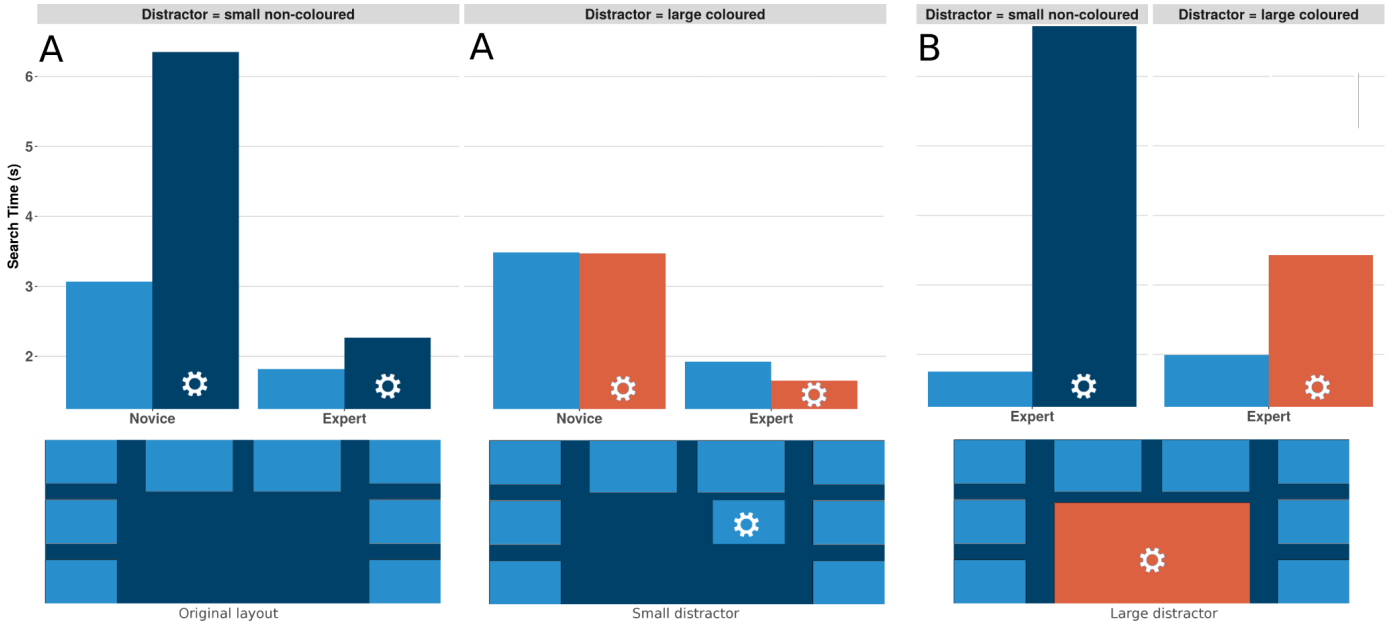


Fig. 10. The effect of adding a distractor on a layout, split between novices and experts. Pane A has a distractor added before the training of the novice model, and pane B has an expert model using a layout with a newly added distractor. Original layout is left, centre is small distractor, and right is large.

6. Application: analysis and improvement of layouts

In this section, we demonstrate how designers can use the model to inform design, either by analysing existing designs or by using it in optimisation of new designs.

6.1. Adding new elements

A common design problem is how to add new elements, such as new buttons, app launchers, or website advertisements, to a layout. As the goal dictates, the designer might want the element to be as salient as possible or to distract as little as possible. The expertise of the user matters also. For example, adding a distracting new element should affect novice and expert users differently. To demonstrate how our model can be used to tackle such questions, we used it to simulate example scenarios in a simple block layout with similarly sized elements, with an added new element, here called a distractor (Fig. 10). In pane Fig. 10a, a distractor is added to a layout as part of an initial design, and a complete novice is then trained to be an expert with it.

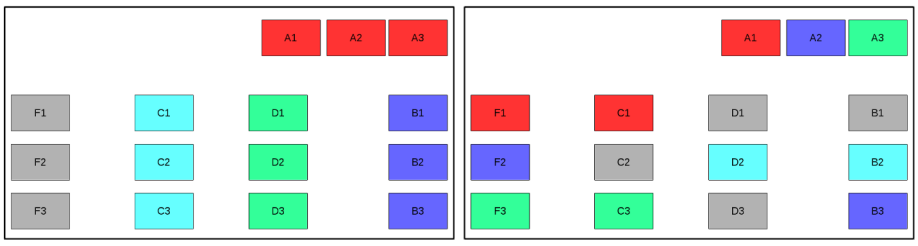
If the distractor is non-salient (left part of Fig. 10a), search times for it are longer for a novice user. However, if the distractor is salient (large and with salient colour), the search for it is faster, though the search time for the target is also impacted negatively. This is because the distractor has high bottom-up saliency. In Fig. 10b, the model is trained to be an expert with a layout lacking the new distractor, and then the distractor is added and the initial impact assessed. Adding a salient distractor, which is easier to find, increases search times for existing targets more than adding a non-salient distractor.

Designers can use the model in cases similar to this example for assessing the trade-off between a new object being noticeable and it being distracting, given the expertise of the user. In the example case, the large coloured distractor is clearly desirable if the goal is to attract users: both novice and experienced users will find the target fairly quickly. This results in slightly higher visual search times for other targets, as the new element distracts users: the increase in novices' search times is about 20% and in experts' is about 10%.

6.2. Grouping by visual feature

Our model of visual search discriminates between targets by their visual features, such as colour. Although it does not distinguish targets by semantic content, it can be used to show that objects that belong together (because of, for example, semantic properties) should be feature-coded similarly. We simulated, as an example, a scenario with a simple grid layout, shown in Fig. 11. It has 15 elements, grouped in 5 groups.

We let the model train to be an expert with the layout and then moved the elements around in the layout. However, the elements within each group were moved such that they were not far from each other. We compared three layouts: one with no coloured objects, one where each group had a distinct colour, and one where the colours (in the same number) were distributed randomly (see Fig. 11). The idea is



(a) Layout grouped by colour. (b) Layout with random colours.

Fig. 11. The two layouts used to inspect grouping by visual feature. The first layout (a) is grouped by colour, and the second (b) has random colours within the groups of three elements. The third layout had the same element placement, but only one colour.

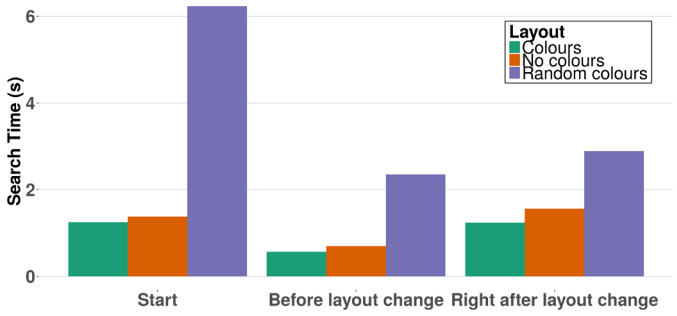


Fig. 12. The effect of colour harmony on layout learnability. The layout with randomly assigned colours is difficult both to learn and relearn in case of layout change.

that the layout with semantically grouped colours should be easier to relearn, because the colours support each other. For example, it is enough for the user to remember that the target was green. Because all green targets are grouped together, the first fixation already brings the user close to the target. However, if the targets are randomly coloured, the green elements being distributed randomly should distract the user from finding the target.

The predicted effect can be seen in Fig. 12. The layout with randomly assigned colours is clearly difficult to use for novices but even for expert users too. The coherently coloured layout, as too the non-coloured layouts, are clearly faster for both user groups. Our model thus supports designs that use the conservative colouring of elements where there are only a few colours to highlight important elements or where colours are used to support the semantic grouping of elements. However, if the users are already familiar with the layout and remember the approximate locations of the elements (e.g., in which group the element belongs), grouping by colour starts to become redundant. At this point, a within-group colour coding might result in the user's ability to skip non-relevant items in a within-group search. To predict this, however, the model would need to have an explicit representation of a group of items. As discussed, modelling how such visual hierarchies might be organised and utilised in layout search and learning is left as a future task.

6.3. Other applications

6.3.1. Saliency editing

As results in pane Fig. 10b demonstrate, visual features of new elements affect how quickly users discover them. A similar design problem is seen when there is an infrequently visited item that users nevertheless must find quickly in the rare case of needing it. Here, the problem is to design a visual object that does not distract the users yet can be quickly found even if the user has no location or feature memory of the object. Our model can be used to evaluate various possible solutions to this problem in a similar fashion to that which is depicted in pane Fig. 10b. A designer must be mindful also of the fact that the relative saliency of two objects depends on the distance between them. With the choice depending on the design goal, a designer might wish to group salient elements together or far from each other (Mateescu and Bajic, 2016).

6.3.2. Less complexity through fewer elements

The number of visual elements in a UI has a clear impact on its learnability, as evident in the higher overall search times for the NYT layout. At some point, a layout becomes too large in practice for a novice to use efficiently. The maximum time allowed for the average novice search depends on the context, but average search times over 30 s, as observed with NYT, are uncomfortably long. Our model can be used for quickly assessing the learnability of layouts, given certain layout elements, and thus evaluating a feasibility of a layout in a given context. Further, the model can be supplied with information about the use frequency of the various elements. This enables a more precise, item-level analysis: which elements can users with various levels of expertise locate? The designer can use the model to fine-tune the number, locations, and features of the layout elements.

6.3.3. Minimum-effort placement

Similarly to the evaluation of new-element placement (see above), the model can be used in the optimisation of the placement of a new object, or even in the arrangement of completely new layouts, given the user history. With the assumption that the user is familiar with an existing layout, the model can predict user behaviour for a new layout. Technically, it is possible also to simulate expertise with more than one layout. The designer could, for example, train the model with multiple popular news sites, then ask where a user with this history would look for the *subscribe* button (Todi et al., 2018). This information can be used in design, evaluation, and optimisation.

7. Design implications

Several implications for the design of graphical layouts can be drawn from the model. While the guidelines we list below are mostly known from previous literature, the model sheds new light on understanding their boundary conditions. Guidelines are traditionally expressed as if-then rules that assume all other things being equal. Computational models permit looking at design decisions under varying circumstances and can yield quantitative predictions that resolve possible trade-off situations. Models can also provide explanations to the guidelines from a theoretical perspective.

1. The fewer elements the better (Granlund et al., 2001). Other things being equal, increasing the number of elements increases visual search time for both novice and experienced users. This effect is produced by two mechanisms in the model. First, for novices the larger number of elements means a larger number of candidate locations to inspect. Second, even for experienced users who know the element locations, there is noise inherent in the involved computations: fixations will under- or overshoot and a completely wrong object will occasionally receive the largest activation.
2. Remove distracting features (Granlund et al., 2001). Distractors that share features with the target are more likely to receive fixations and thereby slow down search.
3. Increase the saliency of important elements (Galitz, 2007). For novice users, objects that are dissimilar to other close objects, receive a large bottom-up activation, and are found more quickly. For expert users, objects that have unique features are easier to locate, even if some objects change their location due to some layout dynamics. The model predicts that visually salient elements are also easier to find after the layout has changed. This is due to their uniqueness in peripherally available features.
4. Follow familiar visual conventions (Galitz, 2007). Users utilise and benefit from long-term memory, including the locations, sizes, and colors of elements. The model predicts that even expert users may spend longer in searching a layout if they initially rely on incorrect recalled information. It is possible to simulate a "prototypical user", who has a prior expectation about the locations and features of known objects. For instance, the model can be trained to expect to find the search bar or icon always on the top of the UI. Building such cultural conventions to the model allow for the testing of how well a new interface conforms to these conventions.
5. Group elements using visually similar features (Galitz, 2007). As seen in the colour harmony example above, random assignment of colour to UI objects distracts search of users across expertise levels. Conversely, colour grouping (or grouping by any other feature) makes the learning of the layout faster, and aids learned users to find targets more quickly.

6. Use visual features sparingly (Galitz, 2007). As seen in the bottom-up computation example, any additional visual feature adds to the activation and, therefore, to the object saliency. Because all objects are compared to others, the total saliency dynamics become very difficult to control by the designer, although the model can be used to predict them.
7. Use white space to distinguish between groups of elements (Galitz, 2007). Because distance lowers the effect of dissimilarity on bottom-up activation and, therefore, object saliency, it is a good idea to distinguish groups of elements from each other using featureless white space. This allows the designer to have more control over the salient objects and the activation dynamics of the UI in general.
8. When changing a layout, avoid moving elements far away from the original location. For an experienced user, the first place to look at is the vicinity of the original location. In particular, non-salient elements are vulnerable to changes that move them far away.

8. Discussion and conclusion

Our results have advanced the understanding of visual learning of graphical layouts. We proposed a new way to integrate bottom-up and top-down control in an adaptive fashion. We believe that the combination of utility learning and feature-guidance is critical to the success of our model in reproducing human data. The combination permits the model to predict differences in how fast a user can learn a layout – or a change in it. Utility learning allows a change in strategy of memory use. As users learn more about the layout, they start relying more on top-down guidance. This is captured by the model. Feature-guidance, on the other hand, describes how visual sampling occurs in the face of limited knowledge and multiple hypotheses on where to look next.

Empirical results are promising. The model shows a good fit with empirical data for search times and patterns with realistic layouts. We tested the model with three layouts from separate domains: a ticket-vending machine, an operating system, and a news website. The model fit was achieved without extensive parameter fitting. We fitted the free parameters universally to all investigated layouts without having to specify layout-dependent parameters. From the full set of 19 parameters, we adjusted only five. The model was able to predict observed change in search times in terms of high linear fit and low prediction error. It also predicted fixation count and the effect of learning on it. Just as importantly, it was able to predict observed search times between different kinds of elements. The model predicted correctly that distinctly coloured elements would be faster to find in the initial learning phase but not show noticeable difference after the layout had been learned.

8.1. Limitations and future work

The parameters of our model can be deemed to reflect an idealised user. This is useful for predicting aggregate results, such as average learning times for a layout. However, often the interest might lie not in average behaviour but in person-to-person differences. While our model has not been tested with such differences in mind, previous research has been successful in modelling user-specific differences in memory (Serna et al., 2007), pointing accuracy (Sarcar et al., 2016), and executive control (Salvucci et al., 2004) (see (Jastrzembski and Charness, 2007) for a review). In future, machine learning methods could be used to find model parameters for individuals (Kangasrääsiö et al., 2019). An individualised model can aid design and optimisation of layouts for individual differences and abilities (Sarcar et al., 2018).

The model can be used to compare frequently and infrequently used objects. In the experiment reported here, all objects had the same frequency. However, the model predicts, for instance, that infrequently utilised visual objects are slower to find, compared to frequently used objects, due to smaller memory activation values which leads to a slower and less reliable recall of positional and feature information. This allows the investigation of more realistic use cases, where the designer can, for example, assess the optimal placement of an object, given the frequency that it is used. On the other hand, infrequently used objects should be placed such that they do not distract the user. However, when these objects are needed, it should be easy to find them even when the user cannot be trusted to remember the location or the features of the object. The model can help the designer balance these contradictory requirements.

As discussed above, with regard to the VSTM decay parameter τ, a more realistic model of visual search would implement low- and high-level search strategies. An example of a low-level strategy is to prefer objects that are close to the currently fixated one (Fleetwood and Byrne, 2006). This minimises saccade time, which depends on the length of the saccade, as well as allows the model to encode close targets without a costly saccade. However, this strategy conflicts with our bottom-up saliency model, wherein the fixation is focused on the target with the most salient object unless guided by further top-down information. More detailed experimental and modelling work would be required to analyse how these two competing processes guide vision. An alternative search strategy is to group the layout into a relatively small number of internally consistent groups and search one group at a time. This helps to maintain a shorter VSTM load, as inhibition of return is broadened to groups instead of objects. However, more research is needed into how the grouping happens, how exactly the groups are stored in working memory and VSTM, and how the within-group search proceeds.

The model was tested on a desktop computer. More work is needed to assess the effects that different display sizes may have. The model has no theoretical limitation that would prevent its application for mobile devices. What changes is the apparent size of elements, which has effects on the utility of peripheral vision. Nevertheless, a validation study with smaller devices should be carried out to confirm the applicability of the model in the mobile domain.

Furthermore, our model does not simulate user interaction with the environment, such as pointing the mouse cursor or the finger at desired target objects. Such work would involve simulating a user, who is tasked with finding and acting on multiple layout elements in the correct order to accomplish a goal. However, it is clear that such a full model of long-term UI interaction will require a model of visual search and learning. To that end, one would need to implement both a pointing model and a task-control model. For example, the model could calculate the movement time from the current cursor location to the target by using a motor-control system similar to that in *EPIC* (Kieras and Meyer, 1997), then add this to the visual search time to simulate user performance. As the model already implements a utility-based control system, this system can be augmented to process task-specific instructions.

More work is needed to make the model more readily actionable in design practice. Due to how the model builds an internal representation of its task interface, it is not possible to directly use the model to analyse images, such as screenshots of interfaces. However, an automated segmenter can be used to transform a UI screenshot into a model-readable file. A prototype version of such an automated segmenter, with the possibility of testing our model with it, is available (Oulasvirta et al., 2018).[1] This allows the quick analysis of visual search times of novice users on any layout. More complicated analyses, such as tweaking object frequencies and setting the users' expertise levels requires modifying the script files of the model. In the future the model could be integrated in design tools, similar to the concept of CogTool (Teo et al., 2012)

[1] https://interfacemetrics.aalto.fi/.

8.2. Conclusion

We have demonstrated, with several examples, how the model can be used to aid in solving design problems related to visual search and layout learning. The model helps designers make decisions about element placement, the number of layout elements, and variations in features (such as the colouring of the elements for saliency or consistency). More generally, the model assists designers by predicting: **1)** visual search times and eye movements, given a layout and a user history; **2)** changes in visual search times and eye movements as the user is exposed to a layout; and **3)** adaptation to dynamically changing layouts.

Our modelling approach is based on the principle of optimal adaptation: our model uses utility learning to find a rationally optimal behaviour, given its resources and the bounds of its architecture and the task environment (Chen and Liu, 2008; Howes et al., 2009). This principle frees the modeller from making assumptions about the low-level strategies, including having to potentially specify different strategies for different tasks and environments. The assumption of optimality, given the resources and limitations, allows for a clear framework where each component of the model can be described in terms of how it assists the agent to achieve the goal, and what bounds it imposes. We conclude that models exploiting reinforcement or utility learning under the idea of bounded rationality offer exciting avenues for applied modelling in HCI.

Declaration of Competing Interest

None.

Acknowledgements

The research was supported by Academy of Finland (grant numbers: 291556, 310947) and European Research Council (grant number: 637991)

Appendix A.

Tables 4, 5 and Figs. 13–19.

Table 4
Cue specific model fit.

Layout	Cue	R^2	*RMSE*
BP	24 h pass	0.874	1.0921
BP	basket content	0.931	0.7512
BP	help	0.429	0.9721
BP	Information	0.79	1.0846
BP	Month for natural	0.705	0.6563
BP	Month for pensioners	0.615	0.5930
BP	Month for pupils	0.563	0.3661
BP	Month for students	0.596	0.3779
BP	Other tickets	0.764	0.3899
BP	single ticket	0.452	0.4691
NYT	blogs	0.731	3.6480
NYT	bussiness	0.882	3.8366
NYT	education	0.143	8.4004
NYT	food	0.796	1.0263
NYT	jobs	0.704	3.6498
NYT	log in	0.476	5.7564
NYT	opinion	0.005	7.4499
NYT	search	0.8	35.0575
NYT	subscribe	0.458	7.3592
NYT	travel	0.809	3.5297
WIN10	audacity	0.889	0.5765
WIN10	camera	0.833	0.5879
WIN10	excel	0.265	0.5728
WIN10	file explorer	0.845	0.6061
WIN10	onedrive	0.772	0.6402
WIN10	outlook	0.768	1.2300
WIN10	paint.net	0.902	1.0736
WIN10	powerpoint	0.742	1.3170
WIN10	store	0.849	0.7557
WIN10	xbox	0.269	1.6822

Table 5
Parameter extrema in parameter exploration.

Parameter	min	max
B_{bl}	0	10
F	0.1	3
f	0.1	3
σ_M	0	1
α	0.01	0.5
σ_U	0.1	1
B_{sa}	0	5
τ	4	100

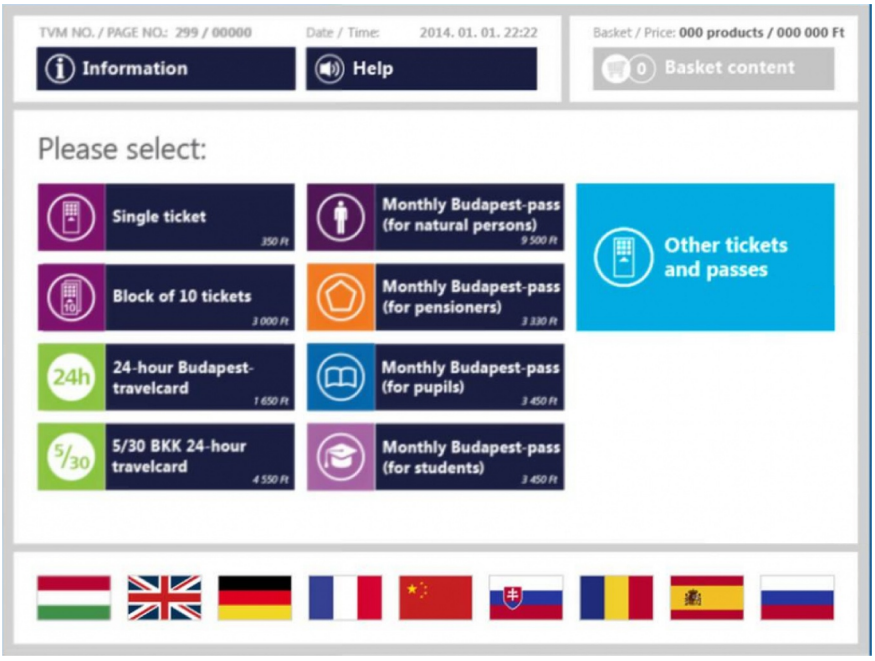


Fig. 13. BP original layout.

Fig. 14. BP modified layout.

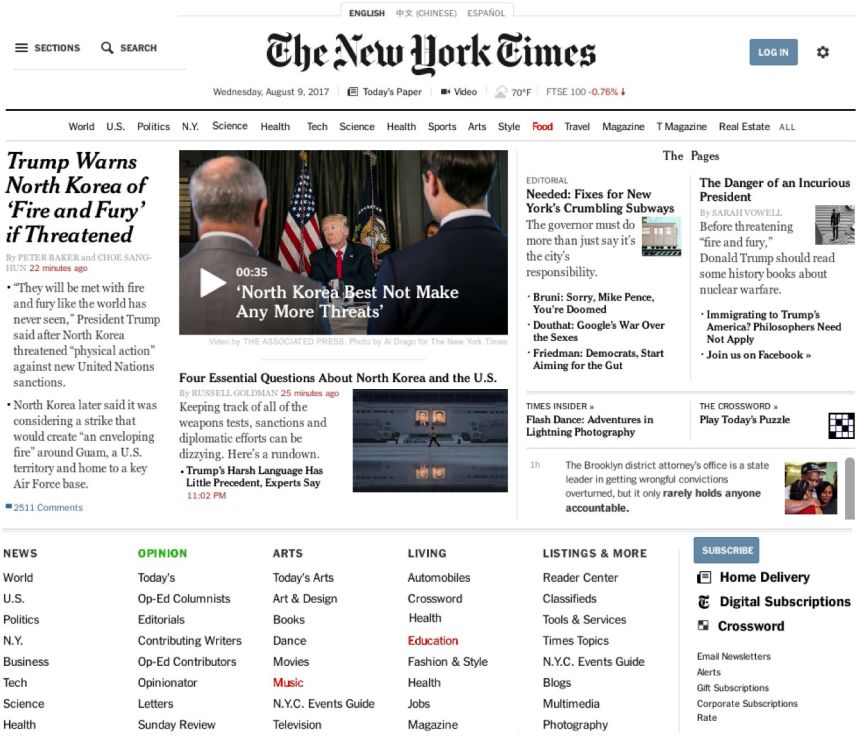


Fig. 15. NYT original layout.

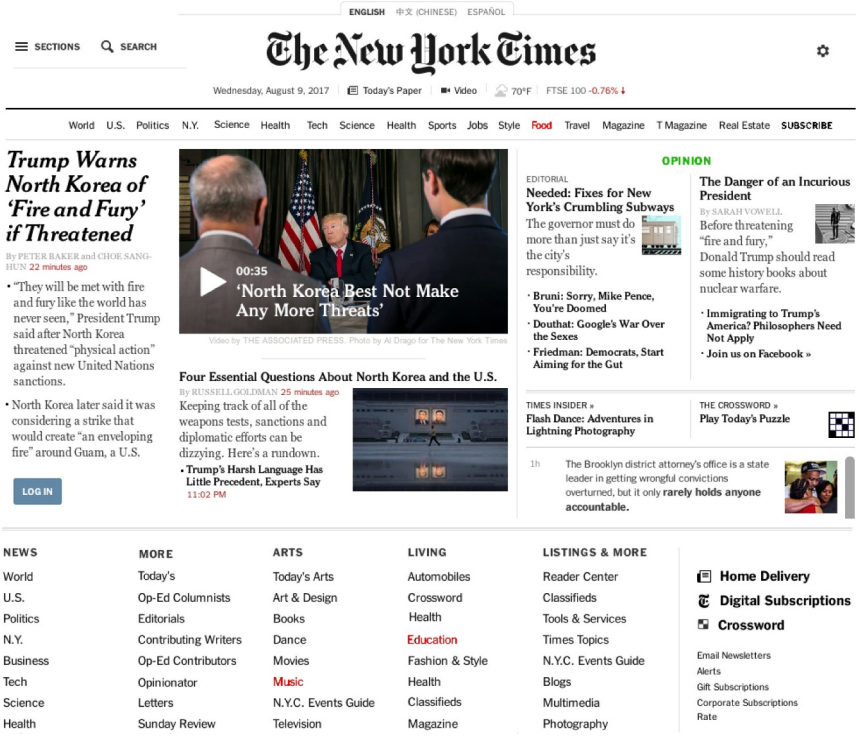


Fig. 16. NYT modified layout.

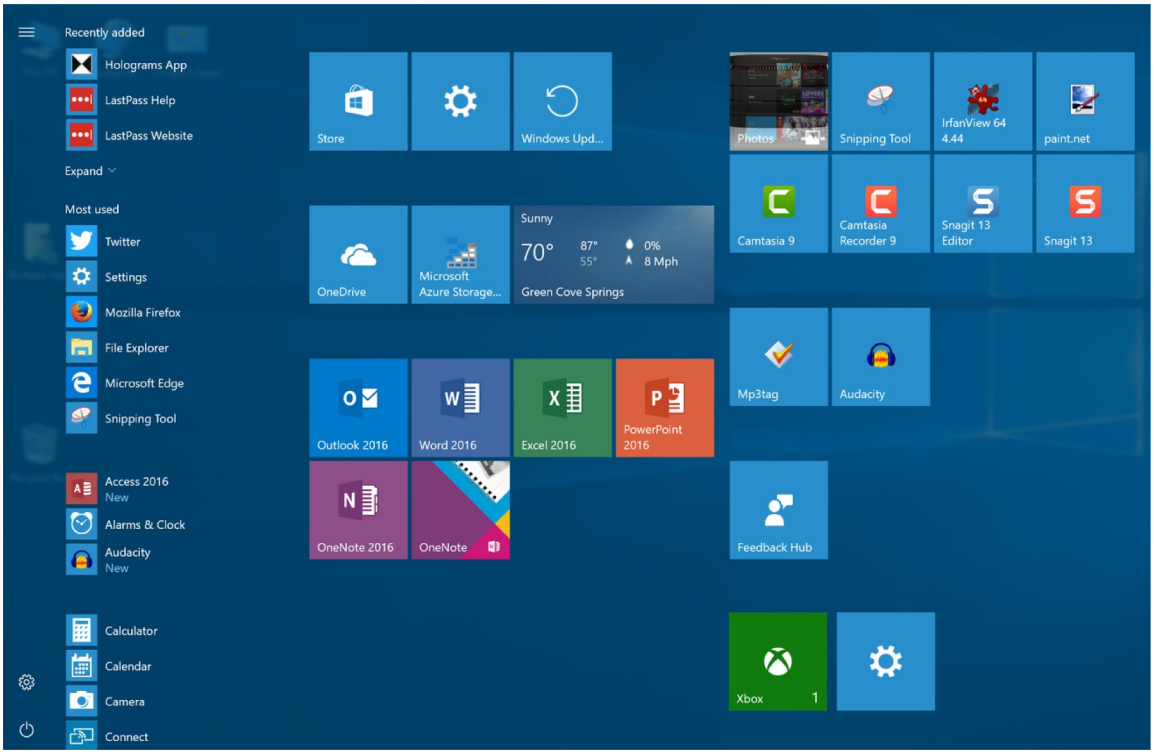


Fig. 17. WIN10 original layout.

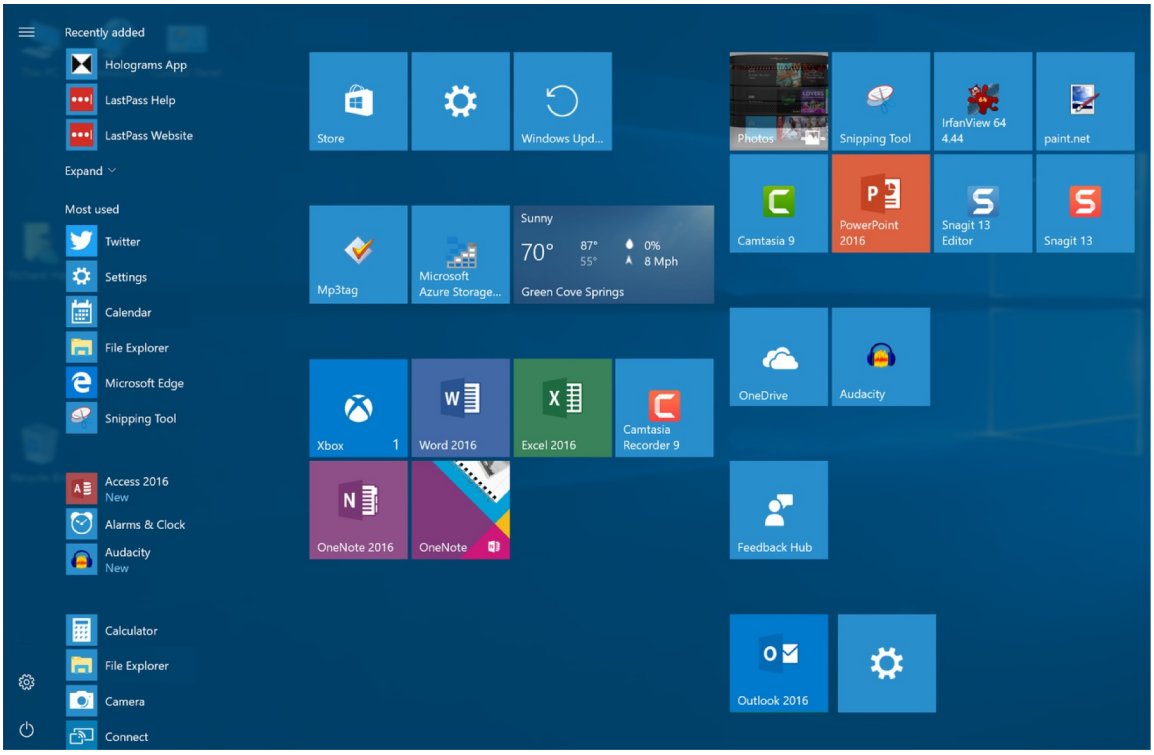


Fig. 18. WIN10 modified layout.

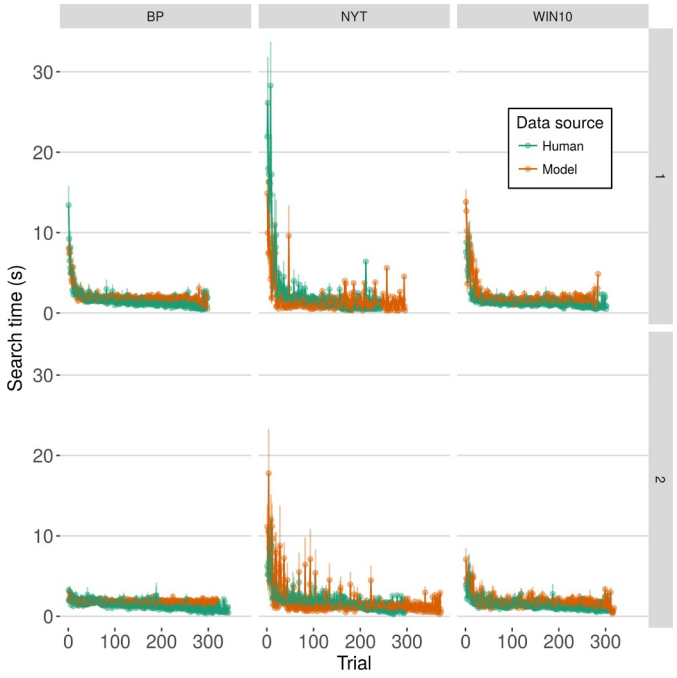


Fig. 19. Model and human RTs aggregated over trials. The vertical panes are the original layout on top, and changed layout on bottom.

References

Anderson, F., Bischof, W.F., 2013. Learning and performance with gesture guides. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 1109–1118.

Anderson, J.R., 1976. Language, Memory and Thought Hillsdale. NJ: LEA.

Anderson, J.R., 2007. How Can the Human Mind Occur in the Physical Universe? Oxford University Press, New York.

Anderson, J.R., 2013. The Architecture of Cognition. Psychology Press.

Anderson, J.R., Bothell, D., Lebiere, C., Matessa, M., 1998. An integrated theory of list memory. J. Mem. Lang. 38 (4), 341–380. https://doi.org/10.1006/jmla.1997.2553.

Bailly, G., Oulasvirta, A., Brumby, D.P., Howes, A., 2014. Model of visual search and selection time in linear menus. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 3865–3874. https://doi.org/10.1145/2556288.2557093.

Bederson, B.B., Russell, D.M., Klemmer, S., 2015. Introduction to online learning at scale. ACM Trans. Comput.-Hum. Interact.(TOCHI) 22 (2), 5.

Borst, J.P., Taatgen, N.A., van Rijn, H., 2015. What makes interruptions disruptive?: A process-model account of the effects of the problem state bottleneck on task interruption and resumption. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 2971–2980.

Byrne, M.D., 2001. ACT-R/PM and menu selection: applying a cognitive architecture to hci. Int. J. Hum.-Comput. Stud. 55 (1), 41–84.

Card, S.K., Moran, T.P., Newell, A., 1983. The Psychology of Human-Computer Interaction. Erlbaum, Hillsdale, NJ.

Cave, K.R., Bichot, N.P., 1999. Visuospatial attention: beyond a spotlight model. Psychon. Bull. Rev. 6 (2), 204–223.

Chen, S.Y., Liu, X., 2008. An integrated approach for modeling learning patterns of students in web-based instruction: a cognitive style perspective. ACM Trans. Comput.-Hum. Interact.(TOCHI) 15 (1), 1.

Chen, X., Bailly, G., Brumby, D.P., Oulasvirta, A., Howes, A., 2015. The emergence of interactive behavior: a model of rational menu search. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 4217–4226.

Chen, X., Starke, S.D., Baber, C., Howes, A., 2017. A cognitive model of how people make decisions through interaction with visual displays. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. ACM, pp. 1205–1216.

Cockburn, A., Gutwin, C., Greenberg, S., 2007. A predictive model of menu performance. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 627–636. https://doi.org/10.1145/1240624.1240723.

Cockburn, A., Gutwin, C., Scarr, J., Malacria, S., 2015. Supporting novice to expert transitions in user interfaces. ACM Comput. Surv. (CSUR) 47 (2), 31:1–31:36.

Dunlop, M., Levine, J., 2012. Multidimensional Pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 2669–2678. https://doi.org/10.1145/2207676.2208659.

Ehret, B.D., 2002. Learning where to look: location learning in graphical user interfaces. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 211–218. https://doi.org/10.1145/503376.503414.

Endress, A.D., Potter, M.C., 2014. Large capacity temporary visual memory. J. Exp. Psychol. 143 (2), 548.

Fleetwood, M.D., Byrne, M.D., 2006. Modeling the visual search of displays: a revised ACT-R model of icon search based on eye-tracking data. Hum.-Comput. Interact. 21 (2), 153–197.

Galitz, W.O., 2007. The Essential Guide to user Interface Design: An Introduction to GUI Design Principles and Techniques. John Wiley & Sons.

Gershman, S.J., Horvitz, E.J., Tenenbaum, J.B., 2015. Computational rationality: a converging paradigm for intelligence in brains, minds, and machines. Science 349 (6245), 273–278.

Granlund, Å., Lafrenière, D., Carr, D.A., 2001. A pattern-supported approach to the user interface design process. International Conference on Human-Computer Interaction: 05/08/2001-10/08/2001.

Howes, A., Lewis, R.L., Vera, A., 2009. Rational adaptation under task and processing constraints: implications for testing theories of cognition and action. Psychol. Rev. 116 (4), 717.

Itti, L., Koch, C., 2000. A saliency-based search mechanism for overt and covert shifts of visual attention. Vision Res. 40 (10), 1489–1506.

Janssen, C.P., Gray, W.D., 2012. When, what, and how much to reward in reinforcement learning-based models of cognition. Cognit. Sci. 36 (2), 333–358. https://doi.org/10.1111/j.1551-6709.2011.01222.x.

Jastrzembski, T.S., Charness, N., 2007. The model human processor and the older adult: parameter estimation and validation within a mobile phone task. J. Exp. Psychol. 13 (4), 224.

John, B.E., Prevas, K., Salvucci, D.D., Koedinger, K., 2004. Predictive human performance modeling made easy. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 455–462. https://doi.org/10.1145/985692.985750.

Jokinen, J.P., Sarcar, S., Oulasvirta, A., Silpasuwanchai, C., Wang, Z., Ren, X., 2017. Modelling learning of new keyboard layouts. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. ACM, pp. 4203–4215.

Judd, C.M., Westfall, J., Kenny, D.A., 2012. Treating stimuli as a random factor in social psychology: a new and comprehensive solution to a pervasive but largely ignored problem. J. Pers. Soc. Psychol. 103 (1), 54.

Kangasrääsiö, A., Jokinen, J.P., Oulasvirta, A., Howes, A., Kaski, S., 2019. Parameter inference for computational cognitive models with approximate Bayesian computation. Cognit. Sci. 43 (6), e12738.

Keating, S., Walker, E., Motupali, A., Solovey, E., 2016. Toward real-time brain sensing for learning assessment: Building a rich dataset. Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems. ACM, pp. 1698–1705.

Kieras, D., 2011. The persistent visual store as the locus of fixation memory in visual search tasks. Cognit. Syst. Res. 12 (2), 102–112.

Kieras, D.E., Hornof, A.J., 2014. Towards accurate and practical predictive models of active-vision-based visual search. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 3875–3884. https://doi.org/10.1145/2556288.2557324.

Kieras, D.E., Meyer, D.E., 1997. An overview of the epic architecture for cognition and performance with application to human-computer interaction. Hum.-Comput. Interact. 12 (4), 391–438.

Kosmyna, N., Tarpin-Bernard, F., Rivet, B., 2015. Adding human learning in

brain–computer interfaces (bcis): towards a practical control modality. ACM Trans. Comput.-Hum. Interact.(TOCHI) 22 (3), 12.
Kowler, E., 2011. Eye movements: the past 25years. Vision Res. 51 (13), 1457–1483.
Kujala, T., Salvucci, D.D., 2015. Modeling visual sampling on in-car displays: the challenge of predicting safety-critical lapses of control. Int. J. Hum.-Comput. Stud. 79, 66–78.
Lewis, R.L., Howes, A., Singh, S., 2014. Computational rationality: linking mechanism and behavior through bounded utility maximization. Top. Cognit. Sci. 6 (2), 279–311.
Malacria, S., Bailly, G., Harrison, J., Cockburn, A., Gutwin, C., 2013. Promoting Hotkey use through rehearsal with ExposeHK. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 573–582.
Mateescu, V.A., Bajic, I.V., 2016. Visual attention retargeting. IEEE MultiMedia 23 (1), 82–91.
Newell, A., 1990. Unified Theories of Cognition. Harvard University Press, Cambridge, Mass.
Newell, A., Rosenbloom, P.S., 1981. Mechanisms of skill acquisition and the law of practice. Cognitive skills and their acquisition. vol. 1. pp. 1–55.
Nyamsuren, E., Taatgen, N.A., 2013. Pre-attentive and attentive vision module. Cognit. Syst. Res. 24, 62–71.
Oulasvirta, A., De Pascale, S., Koch, J., Langerak, T., Jokinen, J., Todi, K., Laine, M., Kristhombuge, M., Zhu, Y., Miniukovich, A., Palmas, G., Weinkauf, T., 2018. Aalto interface metrics (AIM): a service and codebase for computational GUI evaluation. The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings. ACM, New York, NY, USA, pp. 16–19. https://doi.org/10.1145/3266037.3266087.
Paik, J., Kim, J.W., Ritter, F.E., Reitter, D., 2015. Predicting user performance and learning in human–computer interaction with the herbal compiler. ACM Trans. Comput.-Hum. Interact.(TOCHI) 22 (5), 25.
Rayner, K., 2009. The 35th sir frederick bartlett lecture: eye movements and attention in reading, scene perception, and visual search. Q. J. Exp. Psychol. 62 (8), 1457–1506.
Rescorla, R.A., Wagner, A.R., 1972. A theory of pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement. Classical Cond. II 2, 64–99.
Rieman, J., 1996. A field study of exploratory learning strategies. ACM Trans. Comput.-Hum. Interact.(TOCHI) 3 (3), 189–218.
Roberts, S., Pashler, H., 2000. How persuasive is a good fit? A comment on theory testing. Psychol. Rev. 107 (2), 358–367. https://doi.org/10.1037/0033-295X.107.2.358.
Salvucci, D.D., 2001. An integrated model of eye movements and visual encoding. Cognit. Syst. Res. 1 (4), 201–220. https://doi.org/10.1016/S1389-0417(00)00015-2.
Salvucci, D.D., Chavez, A.K., Lee, F.J., 2004. Modeling effects of age in complex tasks: a case study in driving. Proceedings of the 26th Annual Conference of the Cognitive Science Society.
Salvucci, D.D., Macuga, K.L., 2002. Predicting the effects of cell-phone dialing on driver performance. Cognit. Syst. Res. 3 (1), 95–102. https://doi.org/10.1016/S1389-0417(01)00048-1.
Sarcar, S., Jokinen, J.P., Oulasvirta, A., Wang, Z., Silpasuwanchai, C., Ren, X., 2018. Ability-based optimization of touchscreen interactions. IEEE Pervasive Comput. 17 (1), 15–26.
Sarcar, S., Joklnen, J., Oulasvirta, A., Silpasuwanchai, C., Wang, Z., Ren, X., 2016. Towards ability-based optimization for aging users. Proceedings of the International Symposium on Interactive Technology and Ageing Populations. ACM, pp. 77–86.
Sears, A., Jacko, J.A., Chu, J., Moro, F., 2001. The role of visual search in the design of effective soft keyboards. Behav. Inf. Technol. 20 (3), 159–166.
Serna, A., Pigot, H., Rialle, V., 2007. Modeling the progression of Alzheimer's disease for cognitive assistance in smart homes. User Model. User-Adapted Interact. 17 (4), 415–438. https://doi.org/10.1007/s11257-007-9032-y.
Somberg, B.L., 1987. A comparison of rule-based and positionally constant arrangements of computer menu items. ACM SIGCHI Bull. 18 (4), 255–260.
Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction. MIT press Cambridge.
Tatler, B.W., Baddeley, R.J., Gilchrist, I.D., 2005. Visual correlates of fixation selection: effects of scale and time. Vision Res. 45 (5), 643–659.
Teo, L.-h., John, B.E., Blackmon, M.H., 2012. CogTool-Explorer: a model of goal-directed user exploration that considers information layout. ACM, pp. 2479–2488. https://doi.org/10.1145/2207676.2208414.
Todi, K., Jokinen, J., Luyten, K., Oulasvirta, A., 2018. Familiarisation: Restructuring layouts with visual learning models. 23rd International Conference on Intelligent User Interfaces. ACM, pp. 547–558.
Treisman, A., Gelade, G., 1980. A feature integration theory of attention. Cognit. Pschol. 12, 97–136.
Veksler, V.D., Myers, C.W., Gluck, K.A., 2014. SAwSu: an integrated model of associative and reinforcement learning. Cognit. Sci. 38 (3), 580–598. https://doi.org/10.1111/cogs.12103.
Venkatesh, V., Thong, J.Y., Xu, X., 2016. Unified theory of acceptance and use of technology: a synthesis and the road ahead. J. Assoc. Inf. Syst. 17 (5), 328.
Williams, L.G., 1967. The effects of target specification on objects fixated during visual search. Acta Psychol. 27, 355–360.
Wolfe, J.M., 1994. Guided search 2.0 a revised model of visual search. Psychon. Bull. Rev. 1 (2), 202–238.
Wolfe, J.M., 2007. Guided search 4.0: current progress with a model of visual search. In: W, G. (Ed.), Integrated Models of Cognitive Systems, pp. 99–119.
Wolfe, J.M., Horowitz, T.S., 2017. Five factors that guide attention in visual search. Nat. Hum. Behav. 1 (3), 58. https://doi.org/10.1038/s41562-017-0058.