Behav Res (2014) 46:689–701
DOI 10.3758/s13428-013-0414-2

The scene and the unseen: Manipulating photographs for experiments on change blindness and scene memory

Image manipulation for change blindness

Felix Ball · Anne Elzemann · Niko A. Busch

Published online: 6 December 2013

Abstract The change blindness paradigm, in which participants often fail to notice substantial changes in a scene, is a popular tool for studying scene perception, visual memory, and the link between awareness and attention. Some of the most striking and popular examples of change blindness have been demonstrated with digital photographs of natural scenes; in most studies, however, much simpler displays, such as abstract stimuli or "free-floating" objects, are typically used. Although simple displays have undeniable advantages, natural scenes remain a very useful and attractive stimulus for change blindness research. To assist researchers interested in using natural-scene stimuli in change blindness experiments, we provide here a step-by-step tutorial on how to produce changes in natural-scene images with a freely available image-processing tool (GIMP). We explain how changes in a scene can be made by deleting objects or relocating them within the scene or by changing the color of an object, in just a few simple steps. We also explain how the physical properties of such changes can be analyzed using GIMP and MATLAB (a high-level scientific programming tool). Finally, we present an experiment confirming that scenes manipulated according to our guidelines are effective in inducing change blindness and demonstrating the relationship between change blindness and the physical properties of the change and inter-individual differences in performance measures. We expect that this tutorial will be useful for researchers interested in studying the mechanisms of change blindness, attention, or visual memory using natural scenes.

Keywords Change blindness · Change detection · Attention · Scene perception · Image manipulation · Photoshop · Generalized linear modeling

Electronic supplementary material The online version of this article (doi:10.3758/s13428-013-0414-2) contains supplementary material, which is available to authorized users.

F. Ball · A. Elzemann · N. A. Busch (✉)
Institute of Medical Psychology, Charité-Universitätsmedizin Berlin, Luisenstraße 57, 10117 Berlin, Germany
e-mail: niko.busch@charite.de

F. Ball · N. A. Busch
Berlin School of Mind and Brain, Humboldt-Universität zu Berlin, Berlin, Germany

Our subjective experience suggests that we have access to a rich and stable representation of the visual world around us. However, this intuition has been challenged by results obtained with the "change blindness" paradigm (see Jensen, Yao, Street, & Simons, 2011; Rensink, 2002, for reviews). This line of research has demonstrated that observers are often unable to detect large changes in a scene when the change occurs simultaneously with a brief visual disruption, be it a saccade (Grimes, 1996; Hayhoe, Bensinger, & Ballard, 1998; Henderson & Hollingworth, 1999), an eye blink (O'Regan, Deubel, Clark, & Rensink, 2000), a flicker (Rensink, O'Regan, & Clark, 1997), or a distracting stimulus (O'Regan, Rensink, & Clark, 1999). Change blindness has been demonstrated with a wide range of stimuli including artificial scenes with abstract stimuli (Thornton & Fernandez-Duque, 2000; Watanabe, 2003), photographs of natural scenes (Henderson & Hollingworth, 1999; Rensink et al., 1997), video clips (Levin & Simons, 1997), and even in real-life situations (Simons & Levin, 1998). Changes are ecologically highly relevant events, for example in traffic situations: "looking, but failing to see" is one of the key perceptual errors contributing to traffic accidents (Galpin, Underwood, & Crundall, 2009). Thus, a brief distracting event such as the driver's cell phone ringing could make the driver oblivious to a change in the road situation such as a changing traffic light or a pedestrian crossing. Importantly, change blindness does not simply occur when observers fail to look directly at the changing object; in fact, direct fixation of the changing object by no means guarantees change detection: several studies found that observers frequently miss a change even when they are looking directly at it (Caplovitz, Fendrich, & Hughes, 2008; O'Regan et al., 2000). The existence of change blindness obviously points to a limit in our ability to represent, process, and maintain visual scenes, and numerous studies have been conducted to identify the nature of this limitation (Jensen et al., 2011).

In most of the impressive demonstrations of change blindness, digital photographs of natural scenes have been used. In fact, what makes change blindness interesting to the lay person and to many researchers is the conflict between the apparent ease of beholding a natural scene and the surprising failure to detect substantial changes in the scene. Moreover, natural scenes offer a number of advantages over other, more artificial types of displays. First, the incidence and persistence of change blindness is usually higher for changes in natural scenes (Rensink, 2002). Second, change blindness for natural and artificial scenes may involve different processes. For example, EEG studies have demonstrated implicit registration of the change using natural scenes (Fernandez-Duque, Grossi, Thornton, & Neville, 2003), whereas most studies using artificial displays have shown no such effects (Busch, Dürschmid, & Herrmann, 2010; Eimer & Mazza, 2005; Koivisto & Revonsuo, 2003). Furthermore, some of the key factors in change detection are impossible to investigate with artificial displays from the outset. For example, change detection is facilitated for objects considered to be important in the context of the scene (Rensink et al., 1997) or for changes that violate the overall semantic gist of the scene (Hollingworth & Henderson, 2000; Sampanes, Tseng, & Bridgeman, 2008); obviously, no gist is available in artificially created displays. Moreover, researchers may be interested in studying change blindness specifically for certain types of natural scenes—for example, scenes depicting traffic situations (Galpin et al., 2009) or for testing the effect of domain-specific expertise (Werner & Thies, 2000).

However, despite the attractiveness of natural-scene stimuli, the majority of change blindness studies have used artificially generated scenes consisting of simple stimuli such as oriented lines or Gabors (e.g., Thornton & Fernandez-Duque, 2000; Watanabe, 2003) or "free floating" schematic objects (e.g., Busch, 2013; Busch, Fründ, & Herrmann, 2010; Mitroff, Simons, & Levin, 2004). This impression was confirmed by an automated literature survey using PubMed search (www.ncbi.nlm.nih.gov/pubmed; accessed and updated on June 4, 2013) for the keywords "change" + "blindness" + "detection." The search yielded 157 publications, out of which 118 were identified as being visual change blindness experiments.[1] Only 26 of these studies (22 %) used photographs of natural scenes. Among these studies, only 36 different scene images were used on average, and many studies used the identical set of images. Only 15 out of 118 studies (13 %) used images that were created with image-processing software. This finding suggests that researchers face limitations that keep them from using natural-scene stimuli in change blindness experiments.

It is reasonable to ask why natural scenes are not used more frequently in change blindness research. An undeniable advantage of artificial displays is that the experimenter has tight control of stimulus properties—for instance, size, location, and intensity of the change. Another reason might be that natural-scene stimulus material is not as easily produced. To use natural-scene stimuli in a change blindness study, numerous images must be collected and then manipulated, be it by removing, relocating, or exchanging objects in the scene. In the era of cheap digital cameras and searchable online image repositories, image collection is straightforward. This article addresses the second and less trivial task: image manipulation.

In this tutorial, we will explain how to use a freely available image processing software (GNU Image Manipulation Program; GIMP) to manipulate natural-scene stimuli for change blindness experiments. GIMP is a platform-independent, open-source-licensed software with an active community of users and developers. However, the tools introduced in our tutorial have been implemented in image processing software for over two decades (Schewe, 2000) and thus can be found in most other popular software packages, too. In contrast to general-purpose tutorials, we will specifically detail the workflow for creating subtle and quickly producible image modifications for conducting change blindness experiments. We will also address specific challenges related to such modifications. Next to describing how natural scenes can be altered and how three different types of changes (deletion, position, and color) can be produced, we present results from an experiment that used scenes created with these guidelines. We found that these scenes were effective at producing change blindness. In addition, we analyzed the effect of physical image properties on change detection performance, and compared their effect to that of inter-individual differences in performance measures.

Manipulating natural scenes for a change blindness experiment

Image collection

Experimenters may choose to use their own digital camera and take control of factors such as theme, resolution, depth of field, and so forth, themselves. Alternatively, images can be collected rapidly on the internet using Google image search or online image galleries such as Google's Picasa, SmugMug, Photobucket, and Flickr. Note that these websites or the original providers of these images may place restrictions on the use of the images, especially regarding publication or sharing with the scientific community.

[1] One study listed by PubMed was not available to us.

The collected images need to be stored digitally in any of the file formats available for images. Most file formats use some form of image compression to reduce redundancy of the image data and thus reduce the file size. Importantly, lossy compression methods such as JPEG will diminish image quality. Moreover, the compression of manipulated images may lead to undesired distortions of pixels outside the manipulated area. Thus, we recommend storing images using a lossless compression format (or use no compression at all), such as PNG or TIFF.

Working with GIMP

The collected images need to be manipulated to introduce changes. For this tutorial, we used a freely licensed, open-source, and platform-independent image editing software: the GNU Image Manipulation Program (GIMP; version 2.6.11; www.gimp.org/). Further, the techniques presented here can be easily reproduced using most other image editing software packages, including commercial programs (such as Photoshop, Corel Photo-Paint, Serif Photo Plus, or Xara Photo) and freely available programs (such as Paint.NET, PhoXo, Photo Pos Pro, Photo Filtre, or Pixlr).

GIMP's user interface displays three different windows by default: the Toolbox window, the Main window, and a window containing information on the layers, channels, history, and paths (see Fig. 1). We will refer to the latter one as the Layers window. The Toolbox window (red in Fig. 1) lists all tools as icons in the upper part of the window. A table with a description of the tools used for this tutorial can be found in the supplementary materials (http://oszilla.hgs.hu-berlin.de/CB_GIMP/). Tools are used for image manipulation—for example, to select objects, change their orientation, and shift their position. When a tool is selected (e.g., the Pencil tool in Fig. 1), additional information about this tool is displayed in the lower part of the Toolbox window (red shaded area in Fig. 1). Each tool has different parameters that can be adjusted. For example, if the Pencil tool is selected, parameters such as the opacity level or the radius can be set (see the red shaded area in Fig. 1). The image is always displayed in the Main window (green in Fig. 1). The Main window is the workspace, which is used to apply tools and process the images. Information on layers is provided in the Layers window on the right (blue in Fig. 1).

Layers are an important concept in GIMP, as in most image-editing programs. Layers are used to separate different elements of an image, such as the original and the modified versions of a scene, or a particular object extracted from the scene. Layers are hierarchically organized: The top layer represents the foreground of the new image, and the bottom layer represents its background. The order of layers can be arranged manually by "drag and drop." The advantage of using layers is that each layer represents a dedicated workspace that can be manipulated independently of the other layers. After finishing the processing of each layer, all layers are merged to create the final image (see the "Merging Layers" section below).

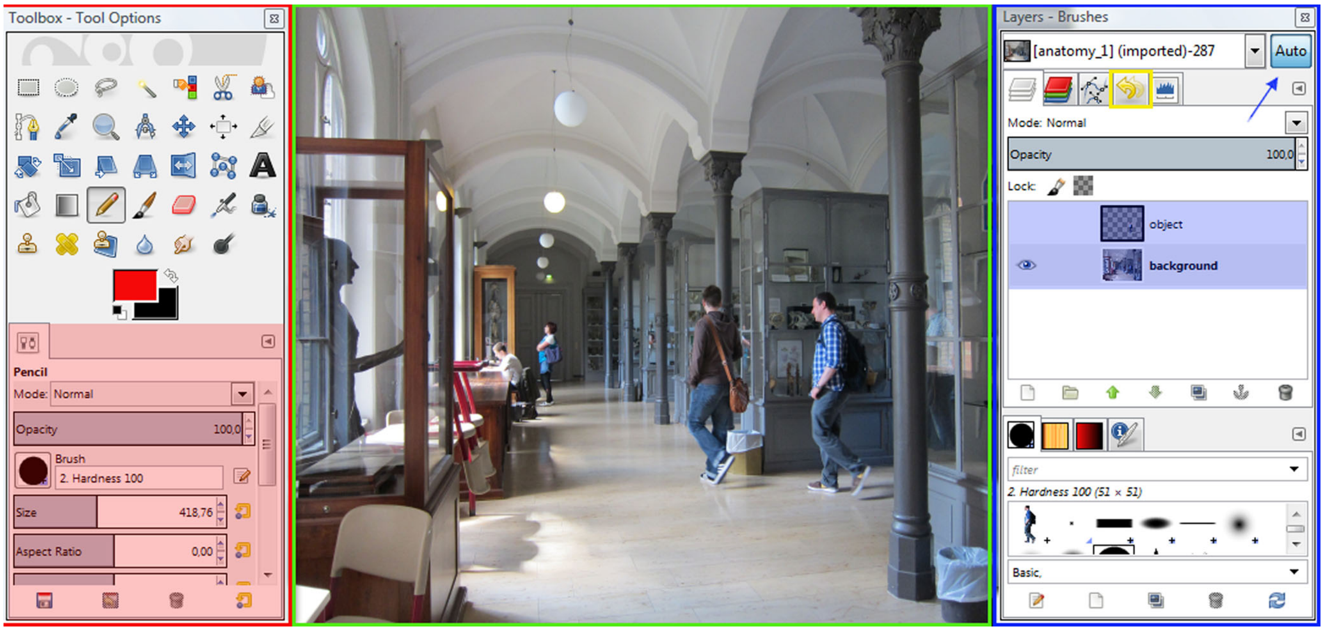
Note that if an image is not displayed in the Layers window after loading, the "Auto" button in the top-right corner of this window (see the arrow in Fig. 1) can be used to display the currently available layers (see the blue-shaded area in Fig. 1). By using the "eye" symbol on the left side of the layer description, layers can be hidden in the Main window (e.g., in Fig. 1 the object layer is hidden, as denoted by a missing "eye" symbol). Hiding different layers in the Main window is especially helpful if only information in one of several layers is being manipulated.

Finally, another useful function is the History dialog (top right tab; the yellow arrow in Fig. 1). The History dialog is part of the Layers window in the program's default settings. Alternatively, it can be accessed in the menu bar under "Windows/Dockable windows/Undo history." It shows all steps of the image processing in a chronological list, starting with opening of the image. Using this history, the user can reset the work to a certain stage of processing by clicking on one of the points on the list. It is important to mention that resetting cannot be applied to specific processing steps in the workflow. For example, only resetting processing steps 20 to 60 out of 500 is not possible. The reset function will always undo all commands from the last processing step to the one of interest. However, it is also possible to redo processing steps that were excluded by clicking on an entry farther down the list. Pressing ctrl+z can be used to undo the most recent action, and ctrl+y can redo the previously undone action (see also "Edit/Undo" and "Edit/Redo" in the menu bar).

We provide sample GIMP files for each of the examples in this tutorial in the supplementary materials (http://oszilla.hgs.hu-berlin.de/CB_GIMP/GIMP files/). For the sake of simplicity, we stored several different layers in each GIMP file (original/modified images and object, shadow and background layers). However, we recommend against storing different versions of the same image in a single GIMP file as long as no back-up of the modified image exists. We and other users experienced problems with reopening such files, which never occurred with files containing a single image.

Choosing an object

The choice of the change object should be guided by several considerations: how easy it will be to detect the change, how much effort the manipulation will require, and how the physical properties of the change will compare to other stimuli in the experiment. The detectability of a change depends on the physical saliency of the change as well as on observers' expectations as to which objects are most likely to change (see Wright, 2005; Ma, Xu, Wong, Jiang, and Hu 2013, see also experiment below). Such expectations may be based on object semantics (e.g., clouds are more likely to change than buildings), guesses about the experimenters intentions, or guesses regarding objects



Pencil
Mode: Normal
Opacity 100,0
Brush 2. Hardness 100
Size 418,76
Aspect Ratio 0,00

Mode: Normal
Opacity 100,0
Lock:
2. Hardness 100 (51 × 51)

Fig. 1 Default layout of GIMP. Important windows are marked by colored squares: the Tool window with tool shortcuts (red) and specific tool information (red-shaded); the Main window (green); and the Layers window (blue) and specific layer information (blue-shaded). The yellow arrow tab opens the History dialog, and the blue arrow points to the "Auto" load button for layers

that lend themselves to a manipulation. As we will describe in detail below, for each scene we noted which objects were selected by experimenters for manipulation and which objects were identified by the observers as most probable change objects. Indeed, observers' expectations matched the experimenters' initial selections with 60 % accuracy. This finding suggests that the first objects that come to mind when choosing change objects are also among the best detectable changes. The expected effort depends to a large extent on the contour and background of the object. Changes are most easily implemented for objects with a clear-cut outline and for homogeneous backgrounds, in which the empty space after deleting the object can be easily filled in. Finally, detectability and physical properties of the change should be as consistent as possible throughout the experiment to avoid confounding experimental conditions. In the "Computing the physical properties of changes" section, we will explain how to measure the physical properties of the change. Sometimes it is desirable to create changes that are difficult to spot. A simple way to create such changes is to alter regions that are usually not interpreted as objects, such as shadows, reflections, and the background (sky, horizon, etc.). However, presenting changes to these regions might cause observers' focus and strategy to differ from natural viewing conditions.

Deletion changes

The objective in this example is to delete the photo from the advertisement on the right pillar in Fig. 2a. This represents an easy example of a deletion change because (1) the object has a very clear-cut outline and (2) the background is a homogeneous surface.

Opening an image

To start working on an image, the image is opened by clicking "File/Open" in the Main window (green in Fig. 1) and rescaled, if necessary. The image is displayed in the Main window and should be listed in the Layers window on the right (blue in Fig. 1). Use the "Auto" button (top-right corner Layers window) if the layer is not listed.

Selecting and deleting an object

To delete an object from the active layer, the object has to be selected using either the Free Select tool (lasso) or the Fuzzy Select tool (magic wand; see the "Color Changes" section). The Free Select tool is based on free-hand mouse selection ("click and drag") and used whenever the object is inhomogeneously colored. It provides the Feather edges option, which smoothens and blends the edges of the selection. The options have to be set individually. We determined a range of 1–5 pixels to be useful, but different images may require different values. In our experience, smaller values improve the selection of small objects and of objects in the image foreground, whereas larger values are better suited for larger objects and objects located in the background. The selection outline (thin line) is formed through several clicks around the object (see Fig. 2c). It is useful to

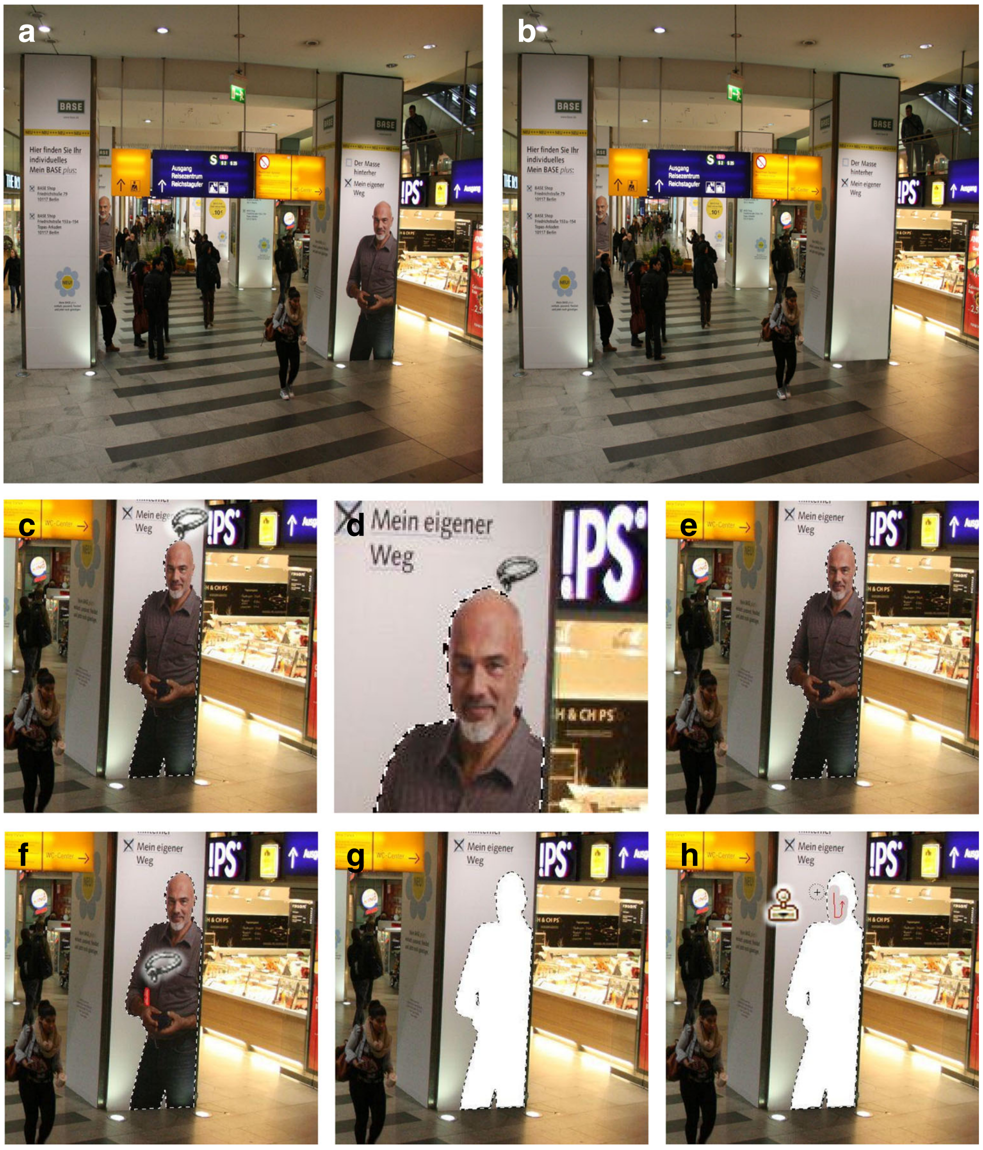


Fig. 2 Creating a deletion change, showing the original (**a**) and modified (**b**) scenes. In this example, the man on the ad is deleted from the original image. The object is selected with the Free Select tool (**c**) and magnified in order to increase selection precision (**d**). After finishing the coarse object selection (**e**), the selection is refined and certain areas are excluded from the selection range (red area in panel **f**). Next, the object is deleted (**g**), and the resulting empty space is filled in using the Clone tool (**h**)

magnify the to-be-deleted object using “ctrl+mouse scroll” to have a better view of its outline and select objects more precisely (see Fig. 2d). After finishing selection by making a double left click (see Fig. 2e), the line surrounding the object will automatically change into a moving dotted line. Importantly, one should not forget to also delete any shadow that belongs to the deleted object.

After the object is selected, the selection can be refined. Areas can be added to the current selection by holding down “shift,” or they can be deselected by holding down the “ctrl” key while left-clicking on the relevant area. When a selection tool is activated, these options can also be found in the lower part of the Toolbox window. In our example, we deselected the previously automatically marked area between the man’s arms and body (see Fig. 2f). Note that as long as an area of the image is selected, all modifications (e.g., with the Clone or Eraser tool) are restricted to this area.

Once the selection process is finished, the objects in the selected area can be deleted by using the “delete” button, leaving an empty space (see Fig. 2g). This empty space needs to be filled in the next step.

Filling in empty space

Homogeneous backgrounds such as plain walls can be easily duplicated and used for filling in the empty space with the Clone tool. This tool requires setting a reference point (“ctrl + left click”), from which information is transferred to the position of the mouse pointer. As can be seen in Fig. 2h, we used the Clone tool to fill in the empty space with a patch of gray surface taken from the pillar. The user can perform “strokes” by holding down the left mouse button and moving the mouse cursor within the empty space. Note that the reference point is not static and will move along with the mouse.

Reference points should be always set close to the empty space, so that the pattern of illumination is kept consistent.

For more complex backgrounds the Clone tool is less useful, because it does not work well with regular patterns such as tiles. Thus, rather than filling the empty space with cloning, the pattern or object can be selected from a suitable area or object in the image using the Fuzzy Select or the Free Select tool, and then copied and pasted into the empty area (see Fig. 3h).

Sometimes the borders of manipulated areas are unnaturally sharp and color or luminance transitions are clearly visible. This problem can be dealt with in different ways. The Clone tool can be used with different levels of opacity (e.g., 20 %) to generate a smooth, gradual color transition. Furthermore, the Blur/Sharpen tool (to blur newly formed edges) or the Smudge tool (to smudge neighboring color areas) can also be used to generate a smooth transition. Alternatively, the Eraser tool with opacity set to a low value can be applied to reduce the intensity around a critical area.

The Pencil tool and the Paintbrush tool can be used to fill small gaps and add small details. Select a matching color and apply it with these tools to a relevant area of the image, hold down "ctrl," and click on that area with the Color Picker tool. These tools were used to fill in the missing part of the truck in the background in Fig. 3i.

Position changes

In this section, we illustrate how to shift the position of a bicyclist in the traffic scene in Fig. 3.

Selecting and moving an object

After loading the image, the object is selected using the Fuzzy Select tool. Once the selection is completed, the selected area can be copied by pressing ctrl+c. Alternatively, the shortcut ctrl+x allows the user to delete the object in the current layer and store it in memory (see Fig. 3c), shown in the lower part of the Layers window (see Fig. 3c and d). With ctrl+v, the object can be pasted into a new layer. This layer will be created automatically and is called a "floating" layer. All "floating" layers have to be named (e.g., "bike" or "shadow"), to avoid automatic merging of layers. Alternatively, new layers can be created by a right click on the Layers window.

If the object casts a shadow (as in our example), it might be helpful to move the object together with its shadow to the new

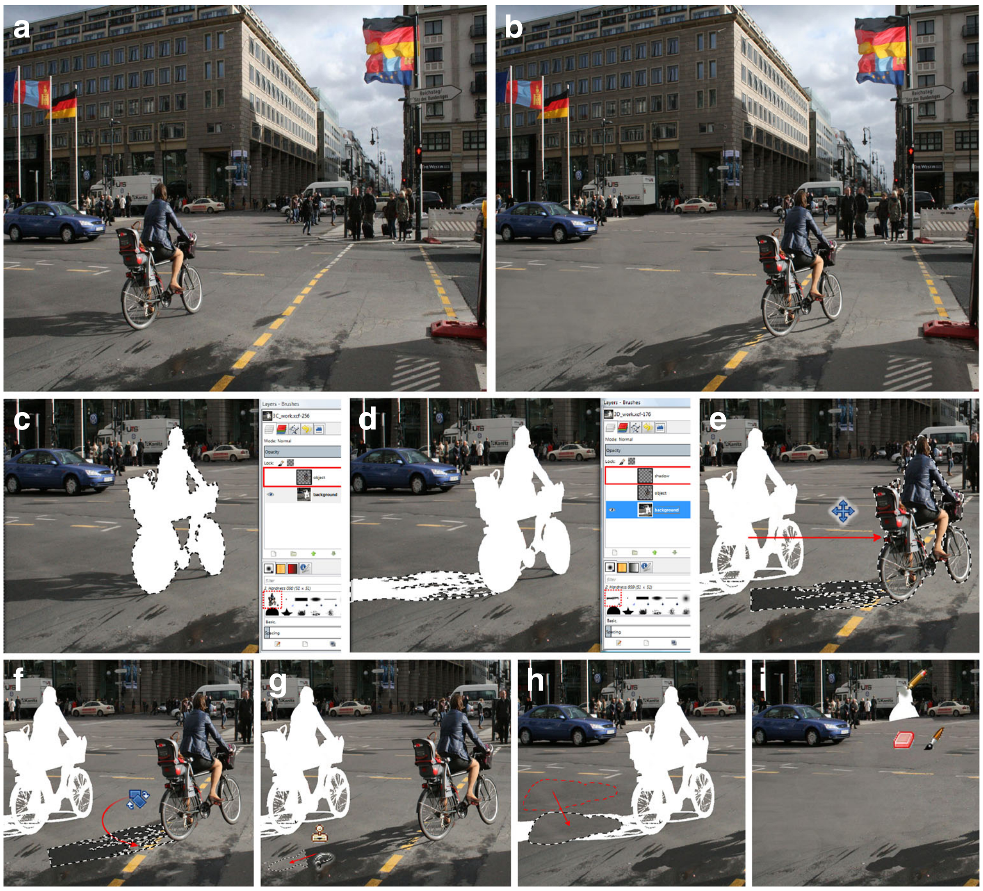


Fig. 3 Creating a position change, showing the original (**a**) and modified (**b**) scenes. In this example, the position of the bicyclist is changed. To shift the position of the bicyclist, the object is selected, extracted, and stored in a new layer (**c**). The shadow can be stored in a separate layer (**d**) but can also be stored together with the bicyclist. Both the bicyclist and her shadow are moved (Moving tool) to their new position (**e**). Next, the shadow is rotated (Rotation tool) to fit its new location (**f**). The missing part of the shadow is outlined with the Free Select tool and filled in with the Clone tool (**g**). Empty space is filled in by "copy and paste" (**h**). The Pencil tool, the Paintbrush tool, and the Eraser tool are then used to add detail (**i**). Merging all layers creates the final image (**b**)

layer. Alternatively, the shadow can also be moved separately, by extracting only the shadow and copying it into its own layer (see Fig. 3d). This offers the possibility to alter the shadow separately.

After selecting the object or shadow layer in the Layers window, the Move tool can be used to move the object/shadow (see Fig. 3e). In this step, it is also possible to change the object's size (Scale tool) or orientation (Rotate tool), mirror the object (Flip tool), or change its apparent angle using the Shear or Perspective tool. Here, we only shifted the bicyclist to a new position and slightly adjusted the orientation of the shadow by using the Rotate tool (see Fig. 3f).

Obscuring location changes

The relocated object often needs to be adjusted to its new location. For example, its shadow might be too dark for the new background. If the shadow is represented in its own layer, its opacity can be adjusted in the Layers window. Alternatively, the Eraser tool with adjusted opacity level can be applied to give the shadow a softer outline. Also, it might be necessary to blur or smooth the object's edges, and one might want to change the color properties of the object in the Brightness–Contrast menu ("Color/Brightness–Contrast") to match its luminance to the new location.

Note that in our example, the shadow was not completely represented on the original image (i.e., the head was missing; Fig. 3f), making it necessary to adjust the shadow after moving it to its new location. To this end, we drew the outline of the missing head with the Free Select tool (Fig. 3g) and then used the Clone tool to fill this selection with a patch of shadow from a different part of the scene (Fig. 3h). Alternatively, the Paintbrush tool can be used to fill in the selected area. The color of the paintbrush can be chosen from the Colors menu (double-click on the color displayed in the Toolbox window) or by choosing a suitable color from within the image itself using the Color Picker tool (hold down "ctrl" + left click). Note that a natural shadow becomes fainter the farther it is from the object. This effect can be achieved easily by lowering the opacity level of the color or by applying the Eraser tool with different opacity levels to the most distant parts of the shadow area.

Merging layers

To create the final image, all layers (object, shadow, background, etc.) need to be merged. Note that layers are merged from top to bottom, like a stack of papers; thus, the foreground object needs to be in the top layer. The order of layers can be changed simply by dragging the different layers in the Layers window. Layers are then merged by selecting the top layer and choosing "Merge down" from the Layers menu.

Color changes

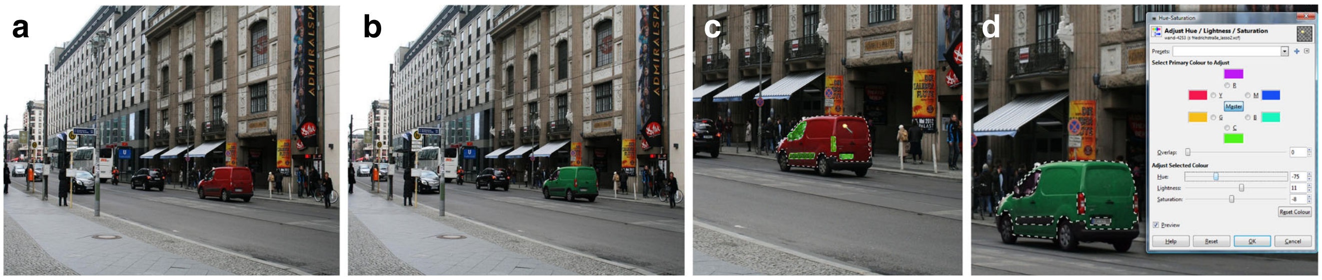
Color changes are the easiest to produce, especially for unicolored objects. If the object that needs to be manipulated is homogeneous in color and distinct from the surrounding background with a clear contour, the object can be easily selected with the Fuzzy Select tool, which automatically selects contiguous areas of similar color (e.g., the red car in Fig. 4a). The user can preset the threshold for color similarity or adjust the threshold during the selection process in the Toolbox window. Note that after selecting the area with a left-click, holding the left mouse button and moving the mouse up or down increases or decreases the selection range. After selecting a suitable object, all parts of the object that are not supposed to be changed should be deselected (we have outlined these areas manually in green in Fig. 4c for better visualization). For example, when changing the color of the car, windows and tires should not be selected and may have to be deselected manually. The hue, saturation, and brightness levels of the selection range can be adapted by using the Colors menu ("Colors/Hue–Saturation"). All changes are displayed online (see Fig. 4d).

Computing the physical properties of changes

One of the disadvantages of using natural-scene stimuli in change blindness experiments is that the physical properties of the change are not as easily controlled and manipulated as with abstract stimuli such as oriented lines or Gabor patches. However, it is desirable to have at least a rough estimate of the physical extent of the change during the image manipulation. This allows one to generate a set of images for which the physical properties of the change are largely consistent across images and, more importantly, between the experimental conditions to which the images are then assigned. Additionally, it may be necessary to analyze relevant properties of the changes at a later point, for example to verify that different experimental conditions are not confounded by differences in these properties, or to analyze directly how they relate to behavioral performance.

GIMP can provide information about change size and luminance difference already during image processing. The histogram summary (under "Colors/Information/Histogram") displays basic properties of the selected areas, such as the number of pixels (change size) and the corresponding mean luminance value. The information provided by the histogram can be used to compare the original and the modified section of the image.

Additionally, scenes can be analyzed conveniently and in an automated fashion using suitable software packages after all image processing is completed. Below, we provide examples for such an analysis written in MATLAB, a general-purpose scientific programming tool (www.mathworks.



Hue-Saturation
Adjust Hue / Lightness / Saturation
wand-4253 (s friedrichstraße_lasso2.xcf)
Presets:
Select Primary Colour to Adjust
R
Y
M
Master
G
B
C
Overlap: 0
Adjust Selected Colour
Hue: -75
Lightness: 11
Saturation: -8
Reset Colour
☑ Preview
Help Reset OK Cancel

Fig. 4 Creating a color change, showing the original (**a**) and modified (**b**) scenes. In this example, the color of the car changes from red to green. The red car in the original scene is selected with the Fuzzy Selection tool (**c**). Windows and tail lights are excluded from selection. We have outlined these areas manually in green for better visualization. Afterward, (**d**) the color of the car is altered using the Colors menu ("Colors/Hue–Saturation") to create the final image

com). Moreover, it may be useful to choose an object (or area of the scene) for image manipulation on the basis of the saliency map of the scene. Saliency is a measure indicating how an object or parts of the scene stand out from the background. The distribution of saliency across the scene is strongly correlated with the pattern of eye movements and the deployment of bottom-up attention. Thus, one might decide to apply changes to less salient parts of the image in order to make the changes harder to detect or control for differences in saliency between experimental conditions. Saliency maps can be computed using the saliency map algorithm provided by Harel, Koch, and Perona (2006).[2] The computational model implemented in this toolbox computes a saliency map based on normalized local feature contrasts within an image. Feature contrasts are defined as local variations of color, brightness, and orientation (Itti, Koch, & Niebur, 1998). We provide a sample script that illustrates how to compute these measures in MATLAB at http://oszilla.hgs.hu-berlin.de/CB_GIMP/Matlab/.

Experiment

In this experiment, we demonstrate that change blindness can be successfully induced by using images manipulated according to the guidelines of this tutorial. We presented three different change types: color, position, and deletion changes. Moreover, we studied the effect of the physical properties of the change (the saliency of the affected area, size of the manipulated area, and luminance difference between the original and modified scenes) on change detection response times.

Moreover, we were interested in the effect of observers' expectations about where the change could occur. Some observers may have expectations about the parts of an image that are easily manipulated or which objects the experimenter might find important to change, and these expectations may influence the deployment of attention. Importantly, change blindness is known to be reduced at the focus of attention (Rensink et al., 1997). Hence, if participants' expectations had an effect on change detection, experimenters would be advised to manipulate less obvious or unexpected objects in a scene. Thus, we asked participants to guess where the change might occur before presenting the changes in order to study the effect of subjective expectations on change detection response times.

The results of the first two analyses can only shed light on how response times depend on a single predictor variable. However, response times might be best explained by a set of predictor variables and their interaction. We were interested in whether the distribution of observed change detection response times in this experiment are explained best by a combination of different physical properties of the change (saliency and size of the manipulated area, and luminance difference between original and modified scene) or by measures of participants' performance (participants' expectation, detection accuracy, and response times). In other words, does the time required to detect a change in a given image depend more on the image's physical properties or on the observer's overall performance? To answer this question, we conducted an analysis using multiple linear regression with different sets of predictor variables to identify which set of predictors results in the most accurate estimate of response times.

Method

Participants

We tested ten observers who volunteered for this experiment (mean age: 26.9 ± 4.3 *SD*; five women, five men; all right-handed). All had normal or corrected-to-normal visual acuity and gave signed informed consent prior to the experiment. The experimental protocol was approved by the ethics committee of the German Psychological Society (DGPS).

Apparatus and stimuli

We collected 34 images of natural scenes using a Canon EOS 400D digital single-lens reflex camera and a Canon ES-F

[2] Available at www.klab.caltech.edu/~harel/share/gbvs.php

Zoom Lens (focal distance 18–55 mm). Some of the scenes were re-used to create 20 modified images each for color, position and deletion changes. Thus, some of the images reappeared during the experiment with a different type of change. The experiment was written in MATLAB (Mathworks Inc.) using the Psychophysics Toolbox (Brainard, 1997; Pelli, 1997). Stimuli were presented on a calibrated 19-in. CRT monitor (1,280 × 1,024 resolution; 100-Hz refresh rate). Participants' head position was stabilized using a chin rest placed 56 cm away from the monitor.

Procedure

Phase I: Rating of change expectation To study the effect of participants' expectations regarding the most likely objects to change, participants completed a short survey prior to the change detection experiment. We presented all of the original scenes, and participants indicated the three locations in each image where they would expect a change by visually marking the locations in the image and by giving verbal descriptions (e.g., "the person with a blue bag"). Participants were advised to rely on their first impression and to choose the objects or regions as quickly as possible. We noted that participants' expectations as to which objects were most likely to change strongly overlapped with the experimenters' preferences as to which objects they would manipulate.

Phase II: Change detection The original (A) and modified scenes (A′) were presented for 600 ms in the sequence A, A′, A′, A, A, A′, A′, . . . , separated by blank displays (400 ms). Participants were asked to press a button as soon as they detected and localized the change, and then to indicate the position of the change using the mouse. If no response was given after 60 display alternations (approximately 2 min), the presentation was terminated and participants had to guess where the change could have occurred. Participants were allowed to make eye movements.

Analysis

In the first analysis, we studied the effect of participants' expectation on response times (number of change presentations). Only correct responses (change was localized correctly) were considered. Reports in the survey were classified as accurate when participants correctly named the changing object or at least one of its parts (e.g., "The jacket of the person on the sidewalk"). We then tested how response times depended on the type of change (deletion, position, or color change) and expectation (accurate vs. inaccurate) using a 3 × 2 factor analysis of variance (ANOVA). The p values were Bonferroni corrected for multiple comparisons (pBF).

A second analysis tested the effect of physical properties of the change on response times (number of change presentations, correct responses only). To this end, for each manipulated image we computed the proportion of changing pixels, the average luminance difference between the original and manipulated images (based on the grayscaled images), and the saliency of the location at which the change occurred. Saliency was computed using the saliency algorithm by Harel et al. (2006), and quantified for subsequent analysis as the average saliency at the location at which the change occurred in both the original and manipulated image. We then computed Spearman correlations (rs) between response times and these three measures.

A third analysis tested which set of predictor variables allowed for the best prediction of response times: is the speed of change detection of a given observer for a given image best predicted from the images physical properties or from the observer's overall performance? To this end, we modeled response times on the basis of a generalized linear regression (`glmfit` function in MATLAB) and tested three different models. The first model (physical properties model) included only physical properties (saliency of the changing area, proportion of changing pixels, average luminance difference) as predictor variables. The second model (performance measure model) included only response measures (accuracy of participants' expectation, change detection success, participants' mean response time) as predictor variables. Finally, a full model included both physical change properties and response measures as predictor variables.

All models were computed using a generalized linear regression (`glmfit` function in MATLAB). Response times served as dependent variable with an inverse Gaussian distribution. Predictors (physical image properties, performance measures, or both) were assumed to be linearly related to the dependent variable (response times linked to incorrect responses were set to 61 change presentations). The success of the three models was evaluated using a leave-one-out cross validation procedure. To this end, the 59 images were divided into a training set of 58 images and a single test image. The linear regression model was computed for the set of training images, and its weights were used to predict the response time for the remaining image. This procedure was repeated 59 times, such that each image served as the test image once, allowing us to compute the difference between predicted and observed response times for each image. The success of each of the three models was then quantified by computing the models' root mean square error (RMSE). Additionally, we computed for each model Pearson correlations (r) between estimated and observed response times.

Results

Participants needed between 8 and 14 presentations (ps) of each change to detect the changing object, indicating that the images in this study induced persistent change blindness. Position changes were detected faster (M_{pos} = 8 ps) than deletion changes (M_{del} = 12.31 ps) or color changes (M_{color} = 13.91 ps), as was indicated by a main effect of change type [$F(2, 18) = 15.695$, $p < .001$, $\eta^2 = .636$]. Change detection was faster when participants had accurate expectations about where the change would occur [expectation: $F(1, 9) = 16.77$, $p = .003$, $\eta^2 = .651$, M_{accur} = 8.98 ps, M_{inacc} = 13.83 ps]. However, this effect was present only for position changes (pBF = .021) and color changes (pBF = .003), but not for deletion changes. This finding was confirmed by a Change Type × Expectation interaction [$F(2, 18) = 5.865$, $p = .011$, $\eta^2 = .395$].

Response times decreased with increasing size of the change ($rs = -.284$, $p = .030$) and with saliency ($rs = -.269$, $p = .040$). Luminance differences had no effect on response times ($rs = -.045$, $p = .737$; see Fig. 5). Similar results were found when outlier images with extreme values of size, saliency, or luminance difference were removed.

By using a generalized linear model, we found that the performance measure model yielded the best estimates of response times (RMSE = 6.58; see Fig. 6). The second-best model was the full model (RMSE = 12.43), followed by the physical properties model (RMSE = 32.67). In line with these findings, we found the highest correlation of the estimated and observed response times for the performance measure model ($r = .89$, pBF < .001) followed by the full model ($r = .57$, pBF < .001). No significant correlation was found for the physical properties model ($r = -.12$, pBF > .05).

Discussion

In this article, we have provided a step-by-step tutorial on how to manipulate digital images for change detection experiments using the free software GIMP. We hope to encourage

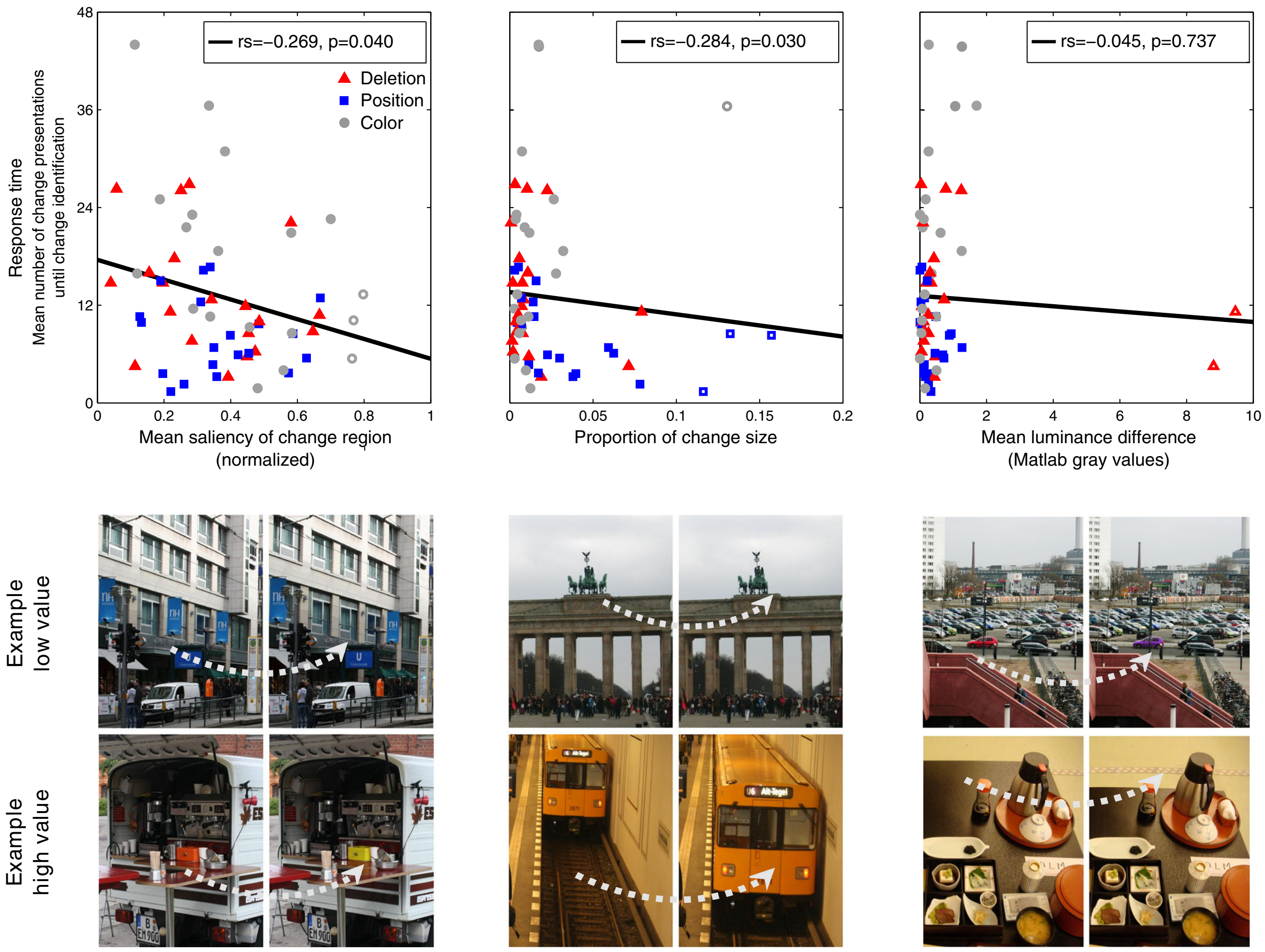


Fig. 5 Spearman correlation plots for image set and physical change properties. Response times were plotted for all change types and the respective physical change property values (top row). Correlations were calculated across change types (solid lines). Solid markers represent the average response times for a given value of a physical change property, and empty markers represent images identified as extreme outliers. Additionally, we have provided two examples for changes with reference to each physical change property. The middle row shows the change with the lowest change property value, and the bottom row shows the change with the highest change property value. The left image in each change property column is always the original image

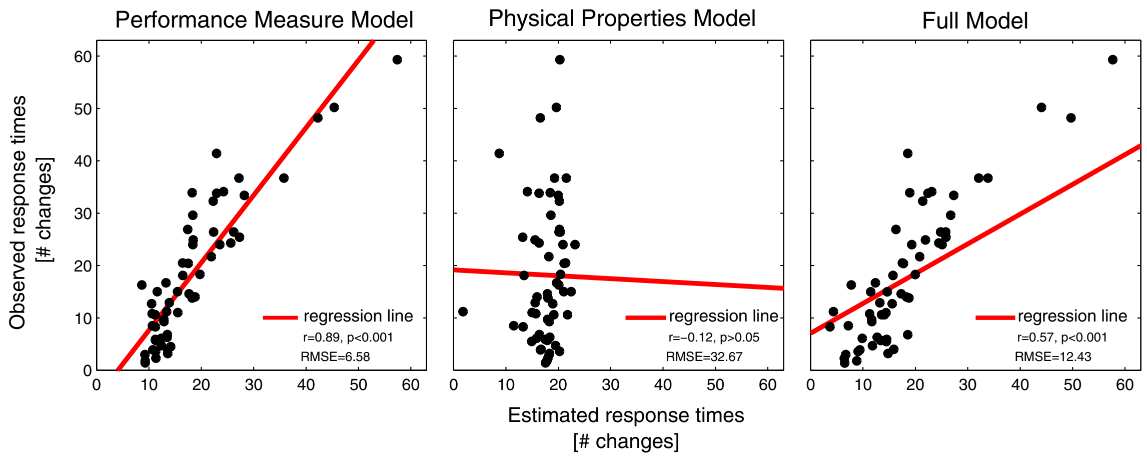


Fig. 6 Observed and estimated response times (averaged across participants), as predicted by the generalized linear model. Response times for a given image are measured as how often a change had to be presented to be detected. The performance measure model (left) predicted response times only on the basis of participants' behavioral performance (participants' expectations, change detection success, and mean response times). The physical properties model (middle) predicted response times only on the basis of the images' physical properties (saliency of the changing area, proportion of changing pixels, and average luminance difference), whereas the full model (right) included both sets of parameters. For illustrative purposes only, two outliers are not shown in the graphs (in the physical properties model, estimated = 233.82, observed = 4.5; in the full model, estimated = 85.36, observed = 4.5)

researchers interested in visual change detection to use natural scenes as stimulus materials, instead of artificial stimuli, and to facilitate the generation of such images. Although the task of manipulating dozens of images may appear daunting, we would point out that by using this tutorial as a guide, GIMP novices were able to reproduce the individual example changes in the tutorial in approximately 15 min. With little training, manipulation of a single image can even be completed in under 10 min (the exact duration depends, of course, on the complexity of the change).

Ma et al., (2013) recently proposed a semiautomatic algorithm that manipulates images to generate changes yielding a user-defined degree of change blindness. The algorithm is based on a computational model of image saliency and complexity. This automated image processing algorithm significantly reduces the effort to generate stimuli for change blindness experiments. However, as mentioned in Ma et al., automated procedures currently face several limitations. First, the algorithm prevents inserting an object in new positions at which it would overlap with other objects. Second, the reconstruction of empty space in the background is largely restricted to areas with homogeneous color and structure. Thus, these procedures are most successfully applied to simple objects with a clear outline that appear in front of a largely homogeneous background. Moreover, the algorithm by Ma et al. (2013) currently does not take into account high-level properties of the change such as image symmetry or semantics. Thus, the procedures for manual image manipulation presented in this article, which do not face the abovementioned limitations, nicely complement recent advances in automatic image processing.

In our experiment, we found that changes that were consistent with observers' expectations as to which objects are most likely to change were detected faster. This finding was expected given the importance of attention for change detection (Rensink et al., 1997). Interestingly, observers expectations strongly overlapped with the experimenters' preference for objects to manipulate. Thus, experimenters may be well-advised to not manipulate the first object in a scene that comes to mind.

Furthermore, we demonstrated that change detection performance was affected by the physical properties of the change: size, luminance difference, and saliency. This finding may not appear particularly surprising, given that change detection is strongly attention-dependent and stronger changes of low-level features are expected to result in stronger engagement of bottom-up attention. However, previous studies have reported inconsistent results regarding the effect of low-level visual features on change detection, with some studies demonstrating that low-level image features have strong effects on change detection (Verma & McOwan, 2010); on the other hand, other studies found only effects of subjective, but not objective, saliency (Wright, 2005), or a stronger effect of top-down, semantic factors (Stirk & Underwood, 2007). Even though the effects of low-level features of the changes may be small or inconsistent, it appears advisable to match images in different experimental conditions according to these properties. A procedure for balancing bottom-up salience in images can be found in Verma and McOwan (2010) and Ma et al. (2013).

We also compared the effect of physical image properties (saliency of the changing area, proportion of changing pixels, average luminance difference) to the effect of participants' overall performance (participants' expectation, change detection success, and mean response time) on response times using a generalized linear model. In other words, we asked whether the time required to detect a change in a given image on a given trial is best predicted by the physical strength of the change (size, saliency, and luminance difference) or by how well the participant performed on other trials. Surprisingly, we found that response times were predicted much better by measures of participants' performance and expectations than image properties, indicating that the trial-to-trial variability in response times that is explained by image properties is small as compared with the variability explained by individual factors. However, we acknowledge that other image properties with potential influence on the speed of change detection were not considered in this experiment, such as the relevance of the changing object for the gist of the scene (Hollingworth & Henderson, 2000; Sampanes et al., 2008; Stirk & Underwood, 2007) or the complexity of the scene.

The existence of change blindness obviously points to a limit in our ability to represent, process, and maintain visual scenes. Just which of the numerous perceptual and cognitive processes involved in change detection are subject to this limit is a matter of ongoing research. Several authors have argued that change blindness results from a limitation during the encoding of visual information (Blackmore, Brelstaff, Nelson, & Troscianko, 1995; O'Regan, 1992; Rensink et al., 1997). Proponents of this view argue that the capacity to represent visual information is strongly limited and suggest that we represent no more than the semantic and structural gist of a scene plus a small portion of the scene's details that are currently at the focus of attention. Alternatively, change blindness could result from a failure to form a stable representation in short-term or long-term memory. Thus, as long as the scene is in view, visual representations may be rich, in line with our phenomenology of a rich experience. However, in the absence of attention, these representations are volatile and are easily overwritten once the original scene disappears and the modified scene is presented (Beck & Levin, 2003; Becker & Pashler, 2002; Landman, Spekreijse, & Lamme, 2003). Of course, limited capacity and representational volatility are not mutually exclusive factors, and some authors have proposed both of them as the cause behind change blindness (see the seminal work by Rensink et al., 1997, and Rensink, 2000). Another cause for change blindness has been demonstrated by Mitroff et al. (2004), who found that sometimes observers are able to recognize both prechange and postchange objects in a subsequent memory test even when they were blind to the changes made to these objects, indicating that change blindness can also result from a failure to compare existing representations of pre- and postchange information. Recently, a number of studies on the role of long-term memory in change detection demonstrated that recognition of the changing objects is often better than change detection performance, suggesting that object memory traces for the changing object are formed, but are often not used in the change detection task (Beck, Peterson, & Angelone, 2007; Hollingworth, 2005; Hollingworth & Henderson, 2002; Varakin & Levin, 2006). Notably, most of the studies on the limits of change detection have used abstract stimuli or sets of "free floating" objects without scene context. Are natural scenes as volatile as displays of oriented bars (Landman et al., 2003), or are they not compared from one moment to the next (Mitroff et al., 2004)? In sum, it is currently unknown whether the same limits apply to the perception of abstract displays and naturalistic stimuli.

Moreover, a number of research questions require the use of naturalistic stimuli, as they are impossible to study with more artificial displays by design. Examples include studies of change detection in traffic situations (Galpin et al., 2009; Koustanaï, Van Elslande, & Bastien, 2012), video camera surveillance (Scott-Brown & Cronin, 2007), studies on the role of expertise in perception (Curran, Gibson, Horne, Young, & Bozell, 2009; Jones, Jones, Smith, & Copley, 2003; Werner & Thies, 2000), or studies on the interplay between bottom-up and top-down processes in scene perception (Stirk & Underwood, 2007). In sum, we believe that the benefits of using natural images for change detection may outweigh the time necessary to create them.

Author Note This work was supported by the German Research Foundation (Grant No. BU2400/1-1). We thank Lyudmyla Kovalenko for comments on an earlier version of the manuscript.

References

Beck, M. R., & Levin, D. T. (2003). The role of representational volatility in recognizing pre- and postchange objects. *Perception & Psychophysics, 65,* 458–468.

Beck, M. R., Peterson, M. S., & Angelone, B. L. (2007). The roles of encoding, retrieval, and awareness in change detection. *Memory & Cognition, 35,* 610–620.

Becker, M. W., & Pashler, H. (2002). Volatile visual representations: Failing to detect changes in recently processed information. *Psychonomic Bulletin & Review, 9,* 744–750.

Blackmore, S. J., Brelstaff, G., Nelson, K., & Troscianko, T. (1995). Is the richness of our visual world an illusion? Transsaccadic memory for complex scenes. *Perception, 24,* 1075–1081.

Brainard, D. H. (1997). The Psychophysics Toolbox. *Spatial Vision, 10,* 433–436.

Busch, N. A. (2013). The fate of object memory traces under change detection and change blindness. *Brain Research, 1520,* 107–115.

Busch, N. A., Dürschmid, S., & Herrmann, C. S. (2010a). ERP effects of change localization, change identification, and change blindness. *NeuroReport, 21,* 371–375.

Busch, N. A., Fründ, I., & Herrmann, C. S. (2010b). Electrophysiological evidence for different types of change detection and change blindness. *Journal of Cognitive Neuroscience, 22,* 1852–1869.

Caplovitz, G. P., Fendrich, R., & Hughes, H. C. (2008). Failures to see: Attentive blank stares revealed by change blindness. *Consciousness and Cognition, 17,* 877–886.

Curran, T., Gibson, L., Horne, J. H., Young, B., & Bozell, A. P. (2009). Expert image analysts show enhanced visual processing in change detection. *Psychonomic Bulletin & Review, 16,* 390–397.

Eimer, M., & Mazza, V. (2005). Electrophysiological correlates of change detection. *Psychophysiology, 42,* 328–342.

Fernandez-Duque, D., Grossi, G., Thornton, I. M., & Neville, H. J. (2003). Representation of change: Separate electrophysiological markers of attention, awareness, and implicit processing. *Journal of Cognitive Neuroscience, 15,* 491–507.

Galpin, A., Underwood, G., & Crundall, D. (2009). Change blindness in driving scenes. *Transportation Research Part F, 12,* 179–185.

Grimes, J. A. (1996). On the failure to detect changes in scenes across saccades. In K. Akins (Ed.), *Perception* (pp. 89–110). Oxford, UK: Oxford University Press.

Harel, J., Koch, C., & Perona, P. (2006). Graph-based visual saliency. In B. Schölkopf, J. C. Platt, & T. Hofmann (Eds.), *Advances in neural information processing systems* (pp. 545–552). Cambridge, MA: MIT Press.

Hayhoe, M. M., Bensinger, D. G., & Ballard, D. H. (1998). Task constraints in visual working memory. *Vision Research, 38,* 125–137.

Henderson, J. M., & Hollingworth, A. (1999). The role of fixation position in detecting scene changes across saccades. *Psychological Science, 10,* 438–443.

Hollingworth, A. (2005). The relationship between online visual representation of a scene and long-term scene memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 31,* 396–411.

Hollingworth, A., & Henderson, J. M. (2000). Semantic informativeness mediates the detection of changes in natural scenes. *Visual Cognition, 7,* 213–235.

Hollingworth, A., & Henderson, J. M. (2002). Accurate visual memory for previously attended objects in natural scenes. *Journal of Experimental Psychology: Human Perception and Performance, 28,* 113–136.

Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20,* 1254–1259.

Jensen, M. S., Yao, R., Street, W. N., & Simons, D. J. (2011). Change blindness and inattentional blindness. *Wiley Interdisciplinary Reviews: Cognitive Science, 2,* 529–546.

Jones, B. T., Jones, B. C., Smith, H., & Copley, N. (2003). A flicker paradigm for inducing change blindness reveals alcohol and cannabis information processing biases in social users. *Addiction, 98,* 235–244.

Koivisto, M., & Revonsuo, A. (2003). An ERP study of change detection, change blindness, and visual awareness. *Psychophysiology, 40,* 423–429.

Koustanaï, A., Van Elslande, P., & Bastien, C. (2012). Use of change blindness to measure different abilities to detect relevant changes in natural driving scenes. *Transportation Research Part F, 15,* 233–242.

Landman, R., Spekreijse, H., & Lamme, V. A. F. (2003). Large capacity storage of integrated objects before change blindness. *Vision Research, 43,* 149–164.

Levin, D. T., & Simons, D. J. (1997). Failure to detect changes to attended objects in motion pictures. *Psychonomic Bulletin & Review, 4,* 501–506.

Ma, L.-Q., Xu, K., Wong, T.-T., Jiang, B.-Y., & Hu, S.-M. (2013). Change blindness images. *IEEE Transactions on Visualization and Computer Graphics, 19,* 1808–1819.

Mitroff, S. R., Simons, D. J., & Levin, D. T. (2004). Nothing compares 2 views: Change blindness can occur despite preserved access to the changed information. *Perception & Psychophysics, 66,* 1268–1281.

O'Regan, J. K. (1992). Solving the "real" mysteries of visual perception: The world as an outside memory. *Canadian Journal of Psychology, 46,* 461–488.

O'Regan, J. K., Deubel, H., Clark, J., & Rensink, R. (2000). Picture changes during blinks: Looking without seeing and seeing without looking. *Visual Cognition, 7,* 191–211.

O'Regan, J. K., Rensink, R. A., & Clark, J. J. (1999). Change-blindness as a result of "mudsplashes". *Nature, 398,* 34.

Pelli, D. G. (1997). The VideoToolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision, 10,* 437–442.

Rensink, R. A. (2000). The dynamic representation of scenes. *Visual Cognition, 7,* 17–42.

Rensink, R. A. (2002). Change detection. *Annual Review of Psychology, 53,* 245–277.

Rensink, R. A., O'Regan, J. K., & Clark, J. J. (1997). To see or not to see: The need for attention to perceive changes in scenes. *Psychological Science, 8,* 368–373.

Sampanes, A. C., Tseng, P., & Bridgeman, B. (2008). The role of gist in scene recognition. *Vision Research, 48,* 2275–2283.

Schewe, J. (2000). 10 years of Photoshop. Retrieved from www.designbyfire.com/pdfs/history_of_photoshop.pdf

Scott-Brown, K. C., & Cronin, P. D. (2007). An instinct for detection: Psychological perspectives on CCTV surveillance*. *Police Journal, 80,* 287–305.

Simons, D. J., & Levin, D. T. (1998). Failure to detect changes to people during a real-world interaction. *Psychonomic Bulletin & Review, 5,* 644–649.

Stirk, J. A., & Underwood, G. (2007). Low-level visual saliency does not predict change detection in natural scenes. *Journal of Vision, 7*, 3, 1–10.

Thornton, I. M., & Fernandez-Duque, D. (2000). An implicit measure of undetected change. *Spatial Vision, 14,* 21–44.

Varakin, D. A., & Levin, D. T. (2006). Change blindness and visual memory: Visual representations get rich and act poor. *British Journal of Psychology, 97,* 51–77.

Verma, M., & McOwan, P. W. (2010). A semi-automated approach to balancing of bottom-up salience for predicting change detection performance. *Journal of Vision, 10*, 3.

Watanabe, K. (2003). Differential effect of distractor timing on localizing versus identifying visual changes. *Cognition, 88,* 243–257.

Werner, S., & Thies, B. (2000). Is "change blindness" attenuated by domain-specific expertise? an expert-novices comparison of change detection in football images. *Visual Cognition, 7,* 163–173.

Wright, M. J. (2005). Saliency predicts change detection in pictures of natural scenes. *Spatial Vision, 18,* 413–430.