Behav Res (2012) 44:439–446
DOI 10.3758/s13428-011-0167-8

The construction of perceptual grouping displays using GERT

Maarten Demeyer · Bart Machilsen

Published online: 20 November 2011
© Psychonomic Society, Inc. 2011

Abstract To study perceptual grouping processes, vision scientists often use stimuli consisting of spatially separated local elements that, together, elicit the percept of a global structure. We developed a set of methods for constructing such displays and implemented them in an open-source MATLAB toolbox, GERT (Grouping Elements Rendering Toolbox). The main purpose of GERT is to embed a contour in a field of randomly positioned elements, while avoiding the introduction of a local density cue. However, GERT's modular implementation enables the user to create a far greater variety of perceptual grouping displays, using these same methods. A generic rendering engine grants access to a variety of element-drawing functions, including Gabors, Gaussians, letters, and polygons.

Keywords Perceptual grouping · Stimulus construction · Contour integration · Texture segmentation · Gabor fields · Local density · GERT

Introduction

Empirical studies on perceptual grouping often employ stimulus displays in which spatially separated local elements are arranged into a specific global configuration. These grouping stimuli come in many flavors: *textured regions* (e.g., Giora & Casco, 2007; Harrison & Feldman, 2009; Julesz, 1981; Lamme, 1995; Nothdurft, 1985; Roelfsema, Lamme, Spekreijse, & Bosch, 2002; Rossi, Desimone, & Ungerleider, 2001; Wolfson & Landy, 1998), *oriented line fields* (e.g., Li & Gilbert, 2002; Tversky, Geisler, & Perry, 2004), *Gabor fields* (e.g., Dakin & Baruch, 2009; Demeyer, De Graef, Verfaillie, & Wagemans, 2011; Field, Hayes, & Hess, 1993; Kovács & Julesz, 1993; Machilsen, Pauwels, & Wagemans, 2009; Nygård, Van Looy, & Wagemans, 2009; Persike & Meinhardt, 2008; Sassi, Vancleef, Machilsen, Panis, & Wagemans, 2010; Watt, Ledgeway, & Dakin, 2008), *dot lattices* (e.g., Bleumers, De Graef, Verfaillie, & Wagemans, 2008; Claessens & Wagemans, 2005, 2008; Kubovy & Wagemans, 1995; Põder, 2011), *Glass patterns* (e.g., Khuu, Moreland, & Phu, 2011; Palomares, Pettet, Vildavski, Hou, & Norcia, 2010; Wilson & Wilkinson, 1998), and many more.

The exact methods used to create these displays are often unique to a particular study or series of studies, too briefly summarized in the method section, or both. To facilitate and standardize stimulus construction in perceptual grouping research, we here present a coherent set of methods and their implementation in the generic framework GERT: the Grouping Elements Rendering Toolbox for MATLAB. Figure 1 illustrates the diversity of image outputs that GERT can deliver in a fast, flexible, and transparent manner.

Although very versatile, GERT is especially well equipped to generate a specific type of stimulus display, where an open or closed contour is embedded in a field of randomly positioned *distractor* or *background* elements, similar to Figs. 1b–e. A difficult problem arises when the research design requires the elimination of *proximity* or *local density* cues in detecting the embedded contour among

Both authors contributed equally to the article and should be considered joint first authors.

M. Demeyer (✉) · B. Machilsen
Laboratory of Experimental Psychology,
University of Leuven (K.U. Leuven),
Tiensestraat 102, bus 3711,
3000 Leuven, Belgium
e-mail: maarten.demeyer@psy.kuleuven.be

B. Machilsen
e-mail: bart.machilsen@psy.kuleuven.be

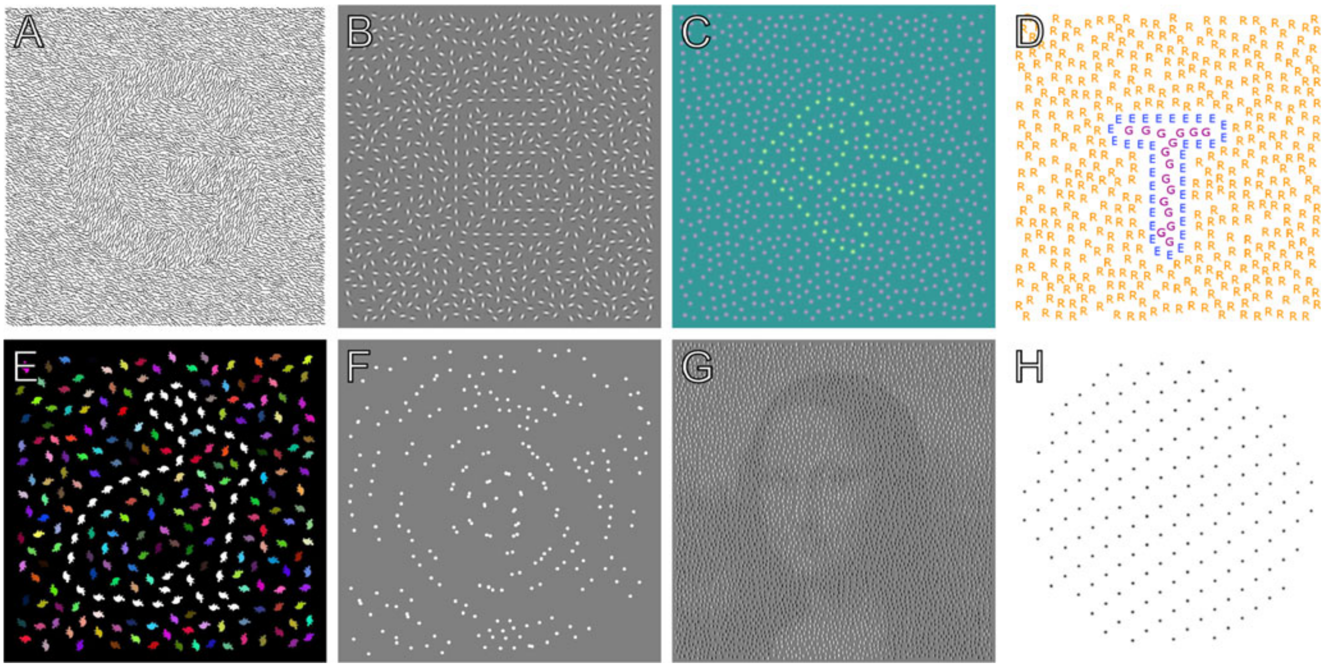

Fig. 1 Example stimuli generated through GERT. A few lines of MATLAB code are sufficient to generate each of these stimuli. **a** Texture-segregation stimulus using oriented line fragments; **b** a letter shape embedded in a field of randomly positioned elements, where Gabor orientation and luminance are manipulated to evoke the shape perceptually; **c** Gaussian blobs on a colored background; **d** Navon-like letter display; **e** shape outline of a hare embedded in a field of small colored tortoises, where both the hare and the tortoises were imported from an SVG file; **f** rotational Glass pattern; **g** randomly positioned Gabor elements with phase offsets determined by the luminance values of an underlying grayscale image; **h** dot lattice consisting of Gaussian blobs

the background elements. GERT implements a comprehensive solution to this problem. In general, the following stimulus construction steps can be identified: (1) Define the underlying contour to be embedded in the display; (2) position local elements on this contour, and populate the remainder of the display with randomly positioned background elements; (3) perform an explicit check for the presence of unwanted proximity cues in the element positions, and minimize them if needed; (4) render the display using a customizable element drawing function. We will describe each of these steps in more detail below.

Step 1: contour definition

First, the global structure to be evoked perceptually (e.g., a shape outline) must be defined. Often, this structure is an open or closed contour, which GERT will define as a discrete set of Cartesian coordinates ordered along the contour (e.g., 1,000 pairs of (x, y)-coordinates). Several methods for obtaining the contour description are available. The coordinate pairs may be read in directly from a plain text file or converted from a Scalable Vector Graphics (SVG) file at a specified resolution. Alternatively, contour definitions may be generated from mathematical parameters; we have currently implemented ellipses and radial frequency patterns (Wilkinson, Wilson, & Habak, 1998). It is also possible to write your own contour generation function or to describe the contour by hand in the MATLAB command window (e.g., use `linspace` to create a line).

Step 2: element placement

We now want to reduce the full contour description to a limited set of *contour* grouping elements, most often embedded in a field of distractor or background elements. We propose and implement a number of methods, for both the contour and the background element placement. Note, however, that for other types of displays, such as dot lattices (Fig. 1h), element positions can also be defined manually, without the use of these specialized GERT routines.

Contour elements

For simple, smooth contours (Fig. 2a), often a clear percept can be evoked by placing the elements equidistant along the contour. In the case of more complex contours, however, this simple algorithm can result in overlap between contour elements. We therefore provide a fast method where such difficult element positions are shifted along the contour to the next suitable position, while equidistant placement is maintained for the other element positions. Figure 2b

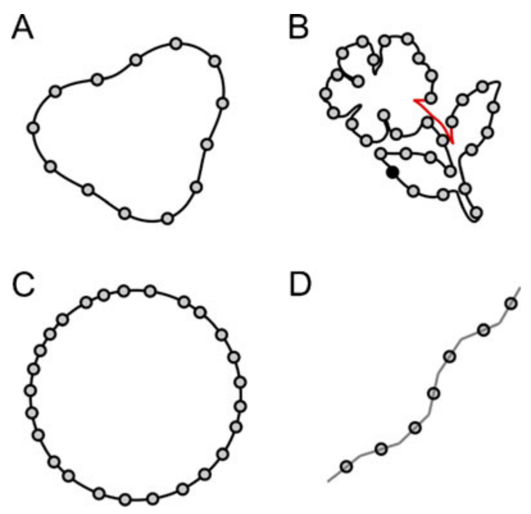

Fig. 2 **a** With smoothly undulating contours, it is easy to position elements at equidistant locations along the contour. **b** If the contour is more complex, equidistant positioning of contour elements results in partial overlap between elements. To avoid overlap with previously placed elements, we shift (red contour segment) difficult elements along the contour. The black element is placed first; the others are placed clockwise along the contour. **c** These elements have been placed at random positions on a circle. No additional elements can be placed without violating a specified minimal distance between the elements. **d** Contour elements can also be positioned on a backbone of connected line segments to form a "snake"

illustrates this for one of the Snodgrass and Vanderwart shape outlines used by Sassi et al. (2010). Positional jitter generated from various types of noise distributions can be applied to both of these methods, either along or perpendicular to the contour. A third GERT method employs random placement of elements on the contour, respecting a minimal Euclidean distance from one another until no more elements can be placed (Fig. 2c). Finally, GERT also implements a procedure for generating "snake" contour elements from an underlying series of connected line segments (Fig. 2d), comparable to the methods used by Hess and Dakin (1999).

Background elements

Traditionally, the most common approach is to place background elements according to an underlying grid and then add positional noise (e.g., Altmann, Deubelius, & Kourtzi, 2004; Bex, Simmers, & Dakin, 2001; Dumoulin & Hess, 2006; Field et al., 1993; Kourtzi, Tolias, Altmann, Augath, & Logothetis, 2003; Kuai & Yu, 2006; Li & Gilbert, 2002; Mathes & Fahle, 2007a, b; Mathes, Trenner, & Fahle, 2006; May & Hess, 2007; Mullen, Beaudot, & McIlhagga, 2000; Nygård, Sassi, & Wagemans, 2011; Tanskanen, Saarinen, Parkkonen, & Hari, 2008). In a similar fashion, Hadad, Maurer, and Lewis (2010) placed background elements on imaginary concentric circles around the embedded closed contour. Some authors also apply a systematic algorithm, rather than random noise, to the initial grid positions (Braun, 1999; Schinkel, Pawelzik, & Ernst, 2005). In their simplest form, algorithms based on a grid can be very fast, but in practice, they often require slow or highly idiosyncratic corrections to produce homogeneous displays. Moreover, the limitations of using a fixed grid can be met when more complex figures are embedded. In some cases, the grid structure may even remain visible in the final stimulus. GERT adheres to a second, more flexible category of methods (e.g., Dakin & Baruch, 2009; Demeyer et al., 2011; Geisler, Perry, Super, & Gallogly, 2001; Kovács & Julesz, 1994; Machilsen & Wagemans, 2011; Verghese, 2009; Watt et al., 2008), where contour elements are first placed as desired and background element positions are then added in an entirely random fashion until the display is full. A minimal distance to previously placed elements is usually the only restriction imposed.

Two approaches can be taken to this random placement. First, the Euclidean distances of previously placed elements to a new randomly generated element position can be computed, to ensure that a minimal distance is kept (e.g., Geisler et al., 2001). The speed of this method does not scale with the resolution at which elements are placed. However, it does slow down with the number of elements already placed in the display, both because the Euclidean distance matrix to compute grows in size and because less "good" positions are available in the display. Moreover, the resulting display is not guaranteed to be filled completely, since some empty spots might never be picked by the random algorithm. A second approach is to keep track of all possible valid positions in the display (i.e., positions respecting a minimal distance from previously placed elements) and to update these valid positions after each element has been placed (e.g., Dakin & Baruch, 2009). This method allows filling the display to the last remaining position, avoiding holes larger than twice the minimal element distance, and will not slow down toward the end. However, the speed of such an algorithm is highly dependent on the resolution of the display.

A hybrid method is implemented in GERT, based on Machilsen et al. (2009). All candidate positions are kept track of. GERT will then generate a whole batch of new random positions (default: 200) and will compute the Euclidean distance matrix between them to eliminate conflicting elements. A simultaneous update of potential future positions with all elements remaining from the batch is then possible. In this way, the size of the Euclidean distance matrix is kept manageable, and the slow process of updating the potential future positions needs to be executed only a handful of times, rather than after each element placement. Speed gains are considerable, especially for high-resolution displays in which many elements need to be placed. For instance, filling an 800 × 800 pixel display with approximately 1,000 elements takes less than 100 ms on a standard desktop computer. In addition, GERT includes methods for restricting background element placement to a specific region of the display and imposing different minimal distances to different sets of existing elements.

For displays such as those shown in Fig. 1a or g, where no explicit contour elements are present, the background element placement routine was used to place *all* elements in the display. To bring forward a perceptual structure, the local element features are then manipulated by applying a different orientation within the figural "g" region and by letting the phase offset of the local Gabor elements depend on the local luminance in a bitmap image of the same size, respectively.

Step 3: minimize local density cues

A lack of consensus

A common and challenging problem is the avoidance of *proximity* or *local density* cues that differentiate contour from background elements, solely on the basis of the relative element spacing. Eliminating this cue is often required to study other grouping cues in isolation. For example, theories of contour integration depend on the collinearity and co-circularity of neighboring Gabor elements (Hess & Dakin, 1999; Watt et al., 2008). An unwanted difference in spacing between contour elements and background elements may then invalidate the conclusions. Many authors (e.g., Braun, 1999; Geisler et al., 2001; Li & Gilbert, 2002; Mathes & Fahle, 2007b; Mullen et al., 2000; Nygård et al., 2009, 2011; Tversky et al., 2004) and even more peer referees warn against local density cues.

It is therefore all the more surprising that few serious or well-documented attempts to quantify local density information have been undertaken. Braun (1999) pointed out that a grid method often results in a different density profile for contour and background elements and suggested an iterative correction algorithm to reduce positional cues. Some researchers (e.g., Mathes & Fahle, 2007a, b; Mathes et al., 2006; May & Hess, 2007) have heeded this warning and have attempted to approximate an equal average distance from contour and background elements to their neighbors, across all generated displays. Other authors have described how they have randomly drawn contour and background elements from a common positional distribution (Nygård et al., 2009, 2011; Watt et al., 2008), applied boundary conditions to an iterative element positioning algorithm (Braun, 1999; Schinkel et al., 2005), or selected stimulus displays on the basis of a subjective screening (Geisler et al., 2001). None of these studies, however, have described a formal test for the presence of a local density cue in a single grouping display.

Controlling the average local density

We have implemented a method for detecting local density differences in GERT. For each element in the display, we calculate its local density as the average distance to the n nearest neighbors. When n is omitted, a Delaunay triangulation is performed to determine the natural neighbors for each element (Mathes & Fahle, 2007b). Elements near the display border are excluded from these computations, since they have lower local density (i.e., fewer neighbors) by definition. We decide on the presence of an *average* local density cue whenever the average of the contour element densities differs significantly from the average of the background element densities. Through performing a statistical test, the severity of the observed difference in average density is weighed by the variability in local densities, as well as the number of elements in the display. When the resulting p-value falls below a threshold α (default: 0.1), the display is said to contain a density cue. In particular, statistical significance is assessed by means of a nonparametric Monte Carlo permutation test. In such a test, the contour and background labels of the elements are randomly reassigned, hundreds to thousands of times. A random baseline distribution of differences in average local density can then be obtained, against which the actually observed difference can be compared. Note that any two arbitrary sets of elements can be used as the input to this local density check. For instance, separate tests may be run for contour versus exterior elements, contour versus interior elements, and interior versus exterior elements. Two alternative local density metrics have additionally been implemented in GERT. The first partitions the display into polygon regions through a Voronoi tessellation, such that all display coordinates within each polygon are closer to the one element enclosed by that polygon than to any other element. The surface area of the polygon is taken as the local density metric. The second alternative metric counts the number of other elements within a radius r around each element.

Whenever the element placement method contains a stochastic component, the presence of a local density cue is in part determined by chance. But it also depends systematically on the relative parameter values of the contour and background element placement functions (e.g., the average spacing along the contour and the minimal spacing in the background). Determining the optimal values is often not an easy problem to solve analytically, as some authors have openly testified (May & Hess, 2007). GERT therefore offers a tool for evaluating a range of parameter values. First, the proportion p^1 of Monte

[1] Note that this is not a p-value in the statistical sense.

Carlo samples for which the difference in average density falls below the actually observed difference is computed for each value in the desired range. GERT then fits a logistic regression to the resulting proportions *p*. The method is illustrated in Fig. 3. A *p* of 1 indicates that the background is always more dense than the contour, whereas a *p* of 0 indicates that the contour is more dense. Where *p* equals .5, the probability that the corresponding element placement parameters will produce a display without a local density cue is highest. This effectively removes the need to try out random parameter values. Note, however, that a local density check should still be performed after each display generation, even when using the "optimal" parameters.

Controlling the full local density distribution

The methods implemented in GERT work well even for very regularly placed contour elements, such as the bear outline in Fig. 3. At the same time, this is a good illustration of the fact that we have, indeed, controlled only for average local density differences, but not for differences in positional regularity. To also eliminate these *variability* cues in the local density information, the *random* method of contour element placement is often preferable (see Fig. 2c). The main parameter of this routine, the minimal distance to previously placed elements, is identical to the main parameter of the background element placement routine. It is therefore trivial to determine the optimal value for each; they should be equal. This does not, however, guarantee that differences do not exist at the level of an individual display. For each of the three local density metrics implemented, a measure sensitive to differences in variability, as well as average distance, is also supplied, on the basis of the distribution of distances to the *n* nearest neighbors for all elements, the distribution of the Voronoi polygon areas, or the distribution of the number of other elements within increasingly larger radii around each element (Braun, 1999). For the two sets of elements that are to be compared, these distributions are then binned, and the absolute difference between the (relative) bin frequencies is summed. A Monte Carlo permutation test similar to the one described above is then run to determine the significance of the observed difference. Whether variability in local density should be controlled for at all is up to the researcher. In some cases, one might prefer to make the embedded contour more easily recognizable by using an equidistant contour element placement, rather than controlling the variability cue (e.g., Sassi et al., 2010).

Step 4: render the display

We have until now been concerned only with computing the positions of individual elements. GERT includes a generic graphical rendering engine to place graphical elements at these positions. The user needs to specify a handle to an *element drawing function* of his choice, as well as the corresponding *drawing parameters* for each element. It is easy to keep certain parameters fixed (pass the value as a scalar), while systematically manipulating or randomizing others (pass the values as a vector). For instance, in Fig. 4, we opted to render previously computed positions as Gabor elements with fixed parameters, except for their orientation, which was manipulated systematically for some elements and randomized for others. Other drawing functions include Gaussian blobs, ellipses, and polygons. To define polygons, SVG files may be read in, allowing their creation in graphical packages such as Inkscape. A Lanczos anti-aliasing filter is applied to avoid jaggedness in polygon element borders. Figure 1e is an example of a display where both the polygonal element and the embedded global contour were imported from SVG files. Should the user require a type of element that is not currently implemented in GERT, it is easy to write a custom element drawing function and plug it into the rendering engine.

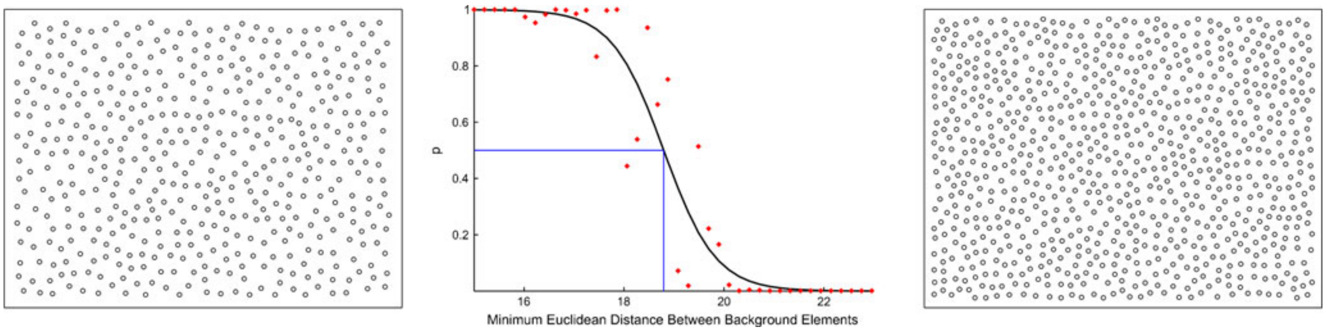

Fig. 3 The shape outline of a bear is visible in the left-hand panel due to a systematic difference in local density between contour and background elements. To remove this proximity cue, we evaluate a range of background element spacing values (middle panel). Using the optimal value at $p = .5$ will render the bear invisible (right-hand panel)

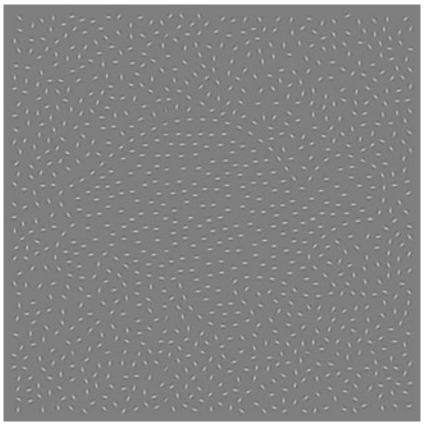

Fig. 4 The shape outline of a tortoise is embedded in a field of approximately 1,000 Gabor elements

We believe that this rendering engine can be of great use even to researchers who do not require any of the previous steps of contour definition and element placement. For instance, the dot lattice in Fig. 1h was created using nothing but the rendering engine and some of the auxiliary GERT functions discussed below. After a MATLAB image matrix is generated, it can easily be imported into a suitable stimulus presentation software package, such as PsychToolbox (Brainard, 1997; Pelli, 1997).

Auxiliary functions

When constructing a stimulus set, the researcher often needs to perform the same elementary operations over and over again. GERT contains additional functionality to speed up the stimulus creation process. For *contour definitions*, the total contour length, center of mass, local tangent lines, and main axis can be computed. The GERT *transform* functions allow the translation, scaling, rotation, and flipping of both contour definitions and element sets. The *plot* functions generate a graphical impression of the data contained within contour definitions and element sets. *Tagging* of elements allows the user to keep track of their status (e.g., *background* or *contour*) and easily retrieve it when needed. *Subsetting and merging* of element sets are supported. In addition, it can easily be determined which elements are located inside and which outside a closed contour. Finally, the automatic *logging* of all GERT activity makes sure that the researcher will never again lose track of the exact parameter values used to create a given stimulus display.

Example stimulus

The example stimuli in the Introduction (Fig. 1) already demonstrated the versatility of GERT. To also illustrate its succinctness and ease of use, we here provide the full MATLAB code to embed the shape outline of a tortoise in an array of Gabor elements (Fig. 4). The contour definition is constructed from an SVG file,[2] and Gabor orientations are manipulated such that exterior elements are oriented randomly, contour elements are collinear to the contour, and interior elements are oriented parallel to the main axis of the shape.

Box 1. MATLAB code to generate the stimulus in Fig. 4

```
% INITIATE GERT
ccc; GERT_Init;

% CONTOUR DEFINITION
C = GERT_GenerateContour_FileSVG('tortoise.svg');
C = GERT_Transform_Center(C,[500 500],'Centroid');

% ELEMENT PLACEMENT
%   ON THE CONTOUR
pec_params.method = 'ParallelEquidistant';
pec_params.cont_avgdist = 30;
[E ors] = GERT_PlaceElements_Contour(C, pec_params);
%   IN THE BACKGROUND
peb_params.min_dist = 24.58;
peb_params.dims = [1 1000 1 1000];
Ea = GERT_PlaceElements_Background(E,[],peb_params);
[f_idx b_idx c_idx] = GERT_Aux_InContour(Ea, E);

% RENDER THE IMAGE
gabel_params.freq      = 0.09;
gabel_params.sigma     = 3;
gabel_params.or(f_idx) = main_axis(C);
gabel_params.or(c_idx) = ors;
gabel_params.or(b_idx) = 2*pi*rand(1,length(b_idx));
IMG = GERT_RenderDisplay(@GERT_DrawElement_Gabor, Ea, gabel_params);
```

In the first line of the code, we clear all variables and initiate the GERT toolbox. We then read in an SVG file to obtain a detailed contour definition *C* of a tortoise and shift its center of mass to the center of a 1,000 × 1,000 pixel display. We ask GERT to place elements at equidistant positions along the contour. This returns an element object *E* and a vector *ors* with the orientations of the local tangent lines to the contour, at the positions of the contour elements. We then fill the remainder of the display with (pseudo-)randomly positioned elements, respecting a minimal Euclidean distance of 24.58 pixels between the element centers. This value minimizes the average local density cue, according to a previously run optimization routine. We also retrieve the indices of the elements that are interior to, exterior to, and exactly on the contour, so that their orientations can be manipulated separately. Because we want all Gabor patches to have the same frequency and standard deviation in the final image, we define these parameter fields as scalars. For the element orientations, a vector equal in length to the number of elements is provided. Should the research design require orientation jitter, this is easily accomplished by adding a `rand(1,E.n)` vector, transformed to the appropriate limits of the uniform noise distribution. The actual rendering is accomplished in the final line of code, where the first input argument is a handle to the drawing function that we want to use.

[2] In fact, from the same SVG file that was used for the local element patches in Fig. 1e.

Conclusions

We have demonstrated that a considerable variety of perceptual grouping displays can be created using GERT, the Grouping Elements Rendering Toolbox for MATLAB. While mainly focused on the embedding of contours in fields of randomly positioned elements, GERT can be used to generate many different types of displays far more rapidly than would be the case if the researcher were to start from scratch. Significant methodological advancements on the topics of element placement and the elimination of local density cues have been presented.

Requirements, license, and web site

MATLAB 2007b or later is required to use GERT, with the Image Processing Toolbox installed. Some options also require the Statistics Toolbox. Basic MATLAB programming skills are assumed. GERT is available as open-source software under the GNU General Public License (Version 3) as published by the Free Software Foundation. A detailed user manual is available at our Web site, http://www.gestaltrevision.be/GERT/, offering a step-by-step introduction to the full functionality of GERT for both novice and advanced programmers. A download link and several example scripts can be found at the same address.

Acknowledgments We thank Lee de-Wit, Naoki Kogo, Krista Overvliet, Tom Putzeys, Michaël Sassi, Kathleen Vancleef, Nathalie Van Humbeeck, and Sander Van de Cruys for testing the GERT toolbox and suggesting improvements. We also thank Geir Nygård and Lizzy Bleumers for laying the foundations from which we started, and Rudy Dekeerschieter for maintaining the GERT web site. Development of the GERT toolbox was supported by a Methusalem grant from the Flemish Government awarded to Johan Wagemans (METH/08/02). Maarten Demeyer is a postdoctoral fellow of the Research Fund K.U.Leuven (PDMK/10/061). Both authors contributed equally to the article and should be considered joint first authors.

References

Altmann, C. F., Deubelius, A., & Kourtzi, Z. (2004). Shape saliency modulates contextual processing in the human lateral occipital complex. *Journal of Cognitive Neuroscience, 16,* 794–804.

Bex, P. J., Simmers, A. J., & Dakin, S. C. (2001). Snakes and ladders: The role of temporal modulation in visual contour integration. *Vision Research, 41,* 3775–3782.

Bleumers, L., De Graef, P., Verfaillie, K., & Wagemans, J. (2008). Eccentric grouping by proximity in multistable dot lattices. *Vision Research, 48,* 179–192.

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

Braun, J. (1999). On the detection of salient contours. *Spatial Vision, 12,* 211–225.

Claessens, P. M. E., & Wagemans, J. (2005). Perceptual grouping in Gabor lattices: Proximity and alignment. *Perception & Psychophysics, 67,* 1446–1459.

Claessens, P. M. E., & Wagemans, J. (2008). A Bayesian framework for cue integration in multistable grouping: Proximity, collinearity, and orientation priors in zigzag lattices. *Journal of Vision, 8*(7, Art. 33), 1–23.

Dakin, S. C., & Baruch, N. J. (2009). Context influences contour integration. *Journal of Vision, 9*(2, Art. 13), 1–13.

Demeyer, M., De Graef, P., Verfaillie, K., & Wagemans, J. (2011). Perceptual grouping of object contours survives saccades. *PLoS ONE, 6,* 1–8.

Dumoulin, S. O., & Hess, R. F. (2006). Modulation of V1 activity by shape: Image-statistics or shape-based perception. *Journal of Neurophysiology, 95,* 3654–3664.

Field, D. J., Hayes, A., & Hess, R. F. (1993). Contour integration by the human visual system: Evidence for a local "association field". *Vision Research, 33,* 173–193.

Geisler, W. S., Perry, J. S., Super, B. J., & Gallogly, D. P. (2001). Edge co-occurrence in natural images predicts contour grouping performance. *Vision Research, 41,* 711–724.

Giora, E., & Casco, C. (2007). Region- and edge-based configurational effects in texture segmentation. *Vision Research, 47,* 879–886.

Hadad, B., Maurer, D., & Lewis, T. L. (2010). The effects of spatial proximity and collinearity on contour integration in adults and children. *Vision Research, 50,* 772–778.

Harrison, S. J., & Feldman, J. (2009). The influence of shape and skeletal axis structure on texture perception. *Journal of Vision, 9* (6, Art. 13), 1–21.

Hess, R. F., & Dakin, S. C. (1999). Contour integration in the peripheral field. *Vision Research, 39,* 947–959.

Julesz, B. (1981). Figure and ground perception in briefly presented isodipole textures. In M. Kubovy & J. R. Pomerantz (Eds.), *Perceptual organization* (pp. 27–55). Hillsdale, NJ: Erlbaum.

Khuu, S. K., Moreland, A., & Phu, J. (2011). The role of shape-from-shading information in the perception of local and global form in Glass patterns. *Journal of Vision, 11*(7, Art. 20), 1–13.

Kourtzi, Z., Tolias, A. S., Altmann, C. F., Augath, M., & Logothetis, N. K. (2003). Integration of local features into global shapes: Monkey and human fMRI studies. *Neuron, 37,* 333–346.

Kovács, I., & Julesz, B. (1993). A closed curve is much more than an incomplete one: Effect of closure in figure–ground segmentation. *Proceedings of the National Academy of Sciences, 90,* 7495–7497.

Kovács, I., & Julesz, B. (1994). Perceptual sensitivity maps within globally defined visual shapes. *Nature, 370,* 644–646.

Kuai, S. G., & Yu, C. (2006). Constant contour integration in peripheral vision for stimuli with good Gestalt properties. *Journal of Vision, 6*(12, Art. 7), 1412–1420.

Kubovy, M., & Wagemans, J. (1995). Grouping by proximity and multistability in dot lattices: A quantitative Gestalt theory. *Psychological Science, 6,* 225–234.

Lamme, V. A. F. (1995). The neurophysiology of figure–ground segregation in primary visual cortex. *Journal of Neuroscience, 15,* 1605–1615.

Li, W., & Gilbert, C. D. (2002). Global contour saliency and local colinear interactions. *Journal of Neurophysiology, 88,* 2846–2856.

Machilsen, B., Pauwels, M., & Wagemans, J. (2009). The role of vertical mirror symmetry in visual shape detection. *Journal of Vision, 9*(12, Art. 11), 1–11.

Machilsen, B., & Wagemans, J. (2011). Integration of contour and surface information in shape detection. *Vision Research, 51,* 179–186.

Mathes, B., & Fahle, M. (2007a). Closure facilitates contour integration. *Vision Research, 47,* 818–827.

Mathes, B., & Fahle, M. (2007b). The electrophysiological correlate of contour integration is similar for color and luminance mechanisms. *Psychophysiology, 44,* 305–322.

Mathes, B., Trenner, D., & Fahle, M. (2006). The electrophysiological correlate of contour integration is modulated by task demands. *Brain Research, 1114,* 98–112.

May, K. A., & Hess, R. F. (2007). Ladder contours are undetectable in the periphery: A crowding effect. *Journal of Vision, 7*(13, Art. 9), 1–15.

Mullen, K. T., Beaudot, W. H. A., & McIlhagga, W. M. (2000). Contour integration in color vision: A common process for the blue–yellow, red–green and luminance mechanisms? *Vision Research, 40,* 639–655.

Nothdurft, H. C. (1985). Orientation sensitivity and texture segmentation in patterns with different line orientation. *Vision Research, 25,* 551–560.

Nygård, G. E., Sassi, M., & Wagemans, J. (2011). The influence of orientation and contrast flicker on contour saliency of outlines of everyday objects. *Vision Research, 51,* 65–73.

Nygård, G. E., Van Looy, T., & Wagemans, J. (2009). The influence of orientation jitter and motion on contour saliency and object identification. *Vision Research, 49,* 2475–2484.

Palomares, M., Pettet, M. W., Vildavski, V. Y., Hou, C., & Norcia, A. M. (2010). Connecting the dots: How local structure affects global integration in infants. *Journal of Cognitive Neuroscience, 22,* 1557–1569.

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

Persike, M., & Meinhardt, G. (2008). Cue summation enables perceptual grouping. *Journal of Experimental Psychology. Human Perception and Performance, 34,* 1–26.

Põder, E. (2011). Perception of multi-stable dot lattices in the visual periphery: An effect of internal position noise. *Journal of Vision, 11*(2, Art. 11), 1–9.

Roelfsema, P. R., Lamme, V. A. F., Spekreijse, H., & Bosch, H. (2002). Figure–ground segregation in a recurrent network architecture. *Journal of Cognitive Neuroscience, 14,* 525–537.

Rossi, A. F., Desimone, R., & Ungerleider, L. G. (2001). Contextual modulation in primary visual cortex of macaques. *Journal of Neuroscience, 21,* 1698–1709.

Sassi, M., Vancleef, K., Machilsen, B., Panis, S., & Wagemans, J. (2010). Identification of everyday objects on the basis of Gaborized outline versions. *i-Perception, 1,* 121–142.

Schinkel, N., Pawelzik, K., & Ernst, U. (2005). Robust integration of noisy contours in a probabilistic neural model. *Neurocomputing, 65–66,* 211–217.

Tanskanen, T., Saarinen, J., Parkkonen, L., & Hari, R. (2008). From local to global: Cortical dynamics of contour integration. *Journal of Vision, 8*((7, Art. 15), 1–12.

Tversky, T., Geisler, W. S., & Perry, J. S. (2004). Contour grouping: Closure effects are explained by good continuation and proximity. *Vision Research, 44,* 2769–2777.

Verghese, P. (2009). Contours in noise: A role for self-cuing? *Journal of Vision, 9*(13, Art. 2), 1–16.

Watt, R., Ledgeway, T., & Dakin, S. C. (2008). Families of models for Gabor paths demonstrate the importance of spatial adjacency. *Journal of Vision, 8*(7, Art. 23), 1–19.

Wilkinson, F., Wilson, H. R., & Habak, C. (1998). Detection and recognition of radial frequency patterns. *Vision Research, 38,* 3555–3568.

Wilson, H. R., & Wilkinson, F. (1998). Detection of global structure in Glass patterns: Implications for form vision. *Vision Research, 38,* 2933–2947.

Wolfson, S. S., & Landy, M. S. (1998). Examining edge- and region-based texture analysis mechanisms. *Vision Research, 38,* 439–446.