Visual Regularity

Peter A. van der Helm

Nijmegen Institute for Cognition and Information
Radboud University Nijmegen

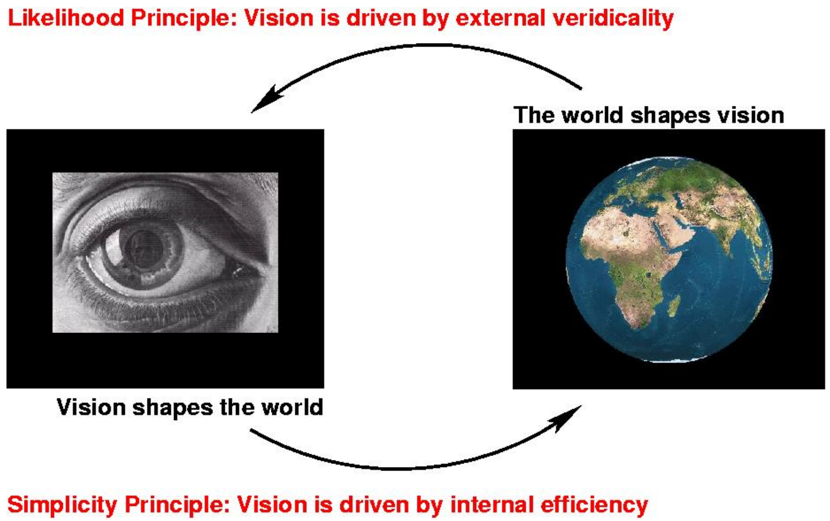
Visual Regularity

Vision and regularities in the world

Regularity detection is integral part of perceptual organization processes.

- Because regularities occur abundantly and are meaningful?
- Because regularities allow for efficient mental representations?

How did vision get sensitive to regularities?



The simplicity and likelihood principles

The likelihood principle (von Helmholtz, 1909)

- "we perceive the most likely* objects or events that would fit the sensory pattern that we are trying to interpret"

  * eg. in terms of frequencies of occurrence in the world

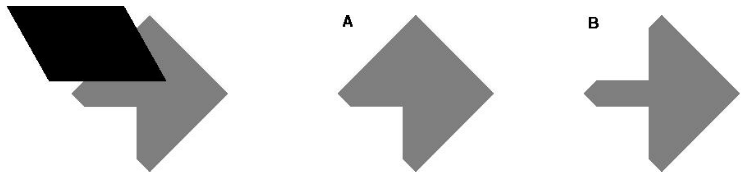
The simplicity principle (Hochberg & McAlister, 1953)

- "the less the amount of information* needed to define a given organization as compared to the other alternatives, the more likely that the figure will be so perceived"

  * "the number of different items we must be given, in order to specify or reproduce a given pattern"

The simplicity-likelihood debate in the 1980s

Do you see a partly occluded grey shape? If so, which shape is it?



Both principles would have predicted A, but for different reasons:

- Simplicity: B has a more complex shape
  (object-centered, ie. viewpoint-independent, factor)
- Likelihood: B would have to take a less likely position
  (viewer-centered, ie. viewpoint-dependent, factor)

The quantifiability of probabilities

The likelihood principle is an appealing learning principle but, applied to vision, the question then is: learning in which space-time world?

In the likelihood paradigm, "world" seems to vary from "the stimulus set during an experiment" to "the universe since the big bang".

However, what is likely in one world may be unlikely in another.

Furthermore, to avoid circular "we see what we see" explanations, the likelihood principle needs objectively quantifiable probabilities.

For instance, one might simulate subjective visual preferences on the basis of frequencies of occurrence of these subjective visual preferences.

This is legitimate but is not the likelihood principle, ie. it is begging the question of how subjective visual preferences come about.

The quantifiability of probabilities

The likelihood principle states that the visual system gives a stimulus the interpretation with the highest probability of being correct.

To this end, vision and vision science would need objective probabilities, eg. in terms of frequencies of occurrence in the world.

Such probabilities, however, are hardly quantifiable, if at all.

Think of Bertrand's Paradox, which shows that probabilities depend on the (in vision to be explained) way in which one structures the world.

Does simplicity precede likelihood?

If probabilities depend on the way in which one structures the world, then one first needs an objective universal method to structure the world.

Such a method is given by the simplicity principle which, moreover, is about perceptual organization, ie. about how vision structures the world.

Hence, it might be that a visual structure's probability is preceded by, or even is to be based on, the complexity of this visual structure.

Thus, the simplicity principle can be seen as a further specification of the likelihood principle – albeit perhaps not as the latter was intended ...

The simplicity and likelihood paradigms

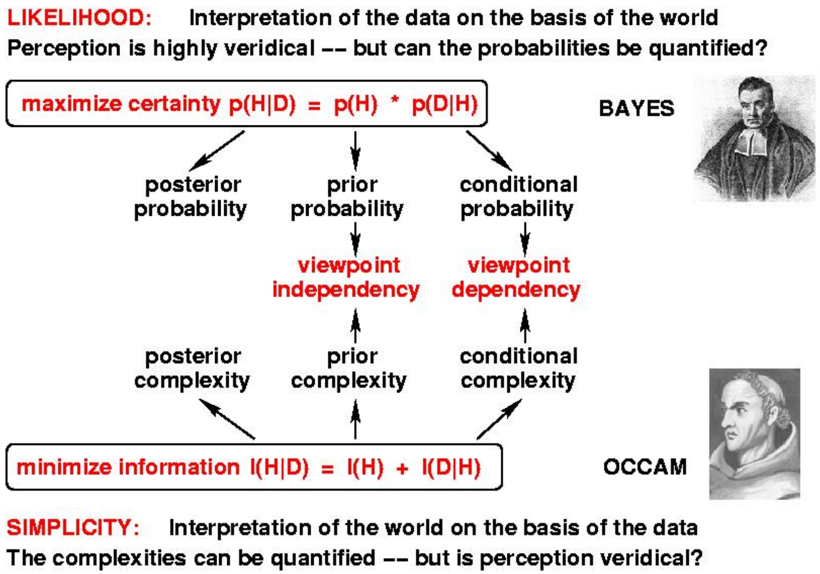
Until the 1970s, the likelihood paradigm focussed on (hardly quantifiable) viewpoint-independent object probabilities.

In the 1970s, the simplicity paradigm provided (better quantifiable) viewpoint-independent object complexities.

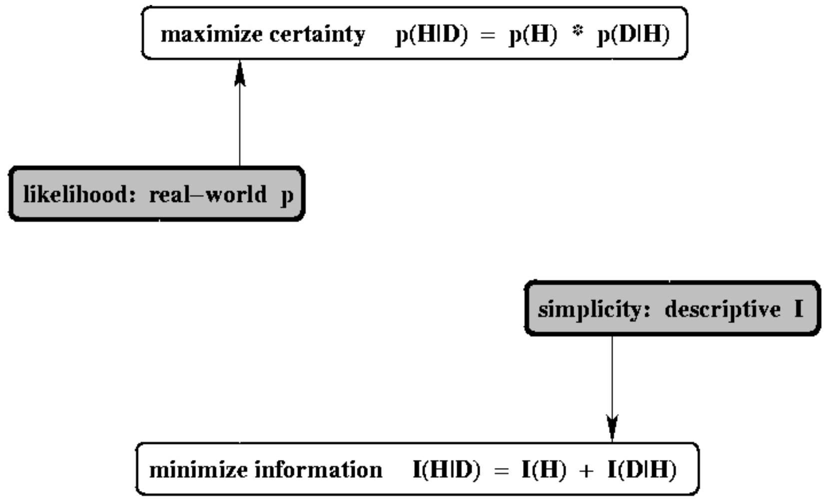
In the 1980s, the likelihood paradigm switched the focus to viewpoint-dependent position probabilities.

Since the 1990s, both paradigms integrate viewpoint-independent factors ("what" priors) and viewpoint-dependent factors ("where" conditionals) in a Bayesian or Occamian way ...

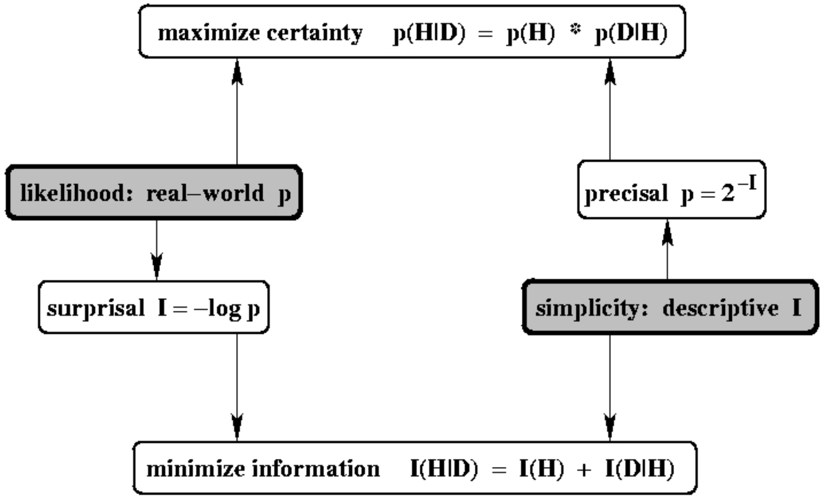
Occam versus Bayes



Principles differ fundamentally ...



… but formulas are equivalent



From surprisals to precisals

The surprisal $I(x) = -\log_2 p(x)$ is Shannon's (1948) ground-breaking quantification of the amount of information (I) in a "message" (x).

It is, however, not based on the content of x, but on the frequency of occurrence (p) of x in the specific world at hand; eg. in the Morse Code, every letter has a code the size of its surprisal in written English.

The precisal $p(x) = 2^{-I(x)}$, on the other hand, is a universal probability (p), based on the descriptive complexity (I) of the content of x.

It agrees with the frequency of occurrence of x in a world filled by a random generator that, each time, first selects a class of objects with the same complexity and only then an object from this class*.

* class size correlates with complexity, so, a particular simple object has a higher probability of being produced than a particular complex object.

The "duality" of simplicity and likelihood

The foregoing shows that the simplicity and likelihood principles, though fundamentally different, imply equivalent optimization formulas.

This formula equivalence enabled deeper research into the differences between complexity-based inference and probability-based inference.

(The latter may yield better predictions about the world but maybe not about vision, and the required probabilities may not be quantifiable.)

Since the 1960s, this question has been studied mainly in:
- the mathematical field of algorithmic information theory (for an overview, see Li & Vitányi, 1997);
- the perceptual field of structural information theory (initiated by Leeuwenberg, 1968).

These theories provided an alternative to Shannon's probabilistic information theory, in a form that can be summarized as follows …

Descriptive versus probabilistic systems

Surprisal codes (based on probabilities)	Simplest descriptive codes (yield precisals)
References to information	Information carriers
One arbitrary nominalistic level	Semantically related nominalistic levels
Length is distribution dependent	Length is fairly language independent
Optimal encoding in one world	Near-optimal encoding in many worlds
Highly veridical in one world	Fairly veridical in many worlds
Special-purpose system (well adapted to one environment)	General-purpose system (fairly adapted to many environments)

Probabilities or complexities?

The foregoing suggests that either principle (ie. simplicity or likelihood) may have guided the evolution of vision.

That is, in general, to prefer either a special-purpose probabilistic system or a general-purpose descriptive system seems a matter of taste:

- if, in a specific world, the required probabilities are quantifiable then, by all means, use them;
- if they are not quantifiable (as eg. in vision), then complexities provide a good alternative.

However, still remaining questions then are:

- can simplest descriptive codes be computed?
- how meaningful are the resulting predictions?

Simplicity requires a pre-defined search space

Simplicity has, in both mathematics and perception, been shown to be a stable concept, but it has no meaning in itself.

It acquires meaning by selecting simplest descriptive codes from a pre-defined search space containing the structures* relevant in the application domain under consideration (here, vision).

* not concrete objects as presupposed in the likelihood paradigm, nor pre-fixed constituents like geons (Binford, 1981) that might do to simulate object recognition (cf. Biederman, 1987), but more general structures that may explain how perception organizes scenes into objects and their constituents.

This search space reflects cognitive constraints imposed by the brain, and determines whether simplest codes are computable.

Structural versus algorithmic information theory

The complexity of an object equals the length of its simplest code.

OK, but how is the search space defined?

Algorithmic information theory (domain independent)

- "simplest codes capture all imaginable regularity"
- triggered a definition of randomness in strings
- useful, but simplest codes of strings then are incomputable

Structural information theory (visual domain)

- "simplest codes capture a maximum of visual regularity"
- requires a definition of visual regularity in strings
- are simplest codes of strings then computable?

Relevance: Conclusion

A general-purpose predictive system based on simplest descriptive codes is a good alternative to special-purpose predictive systems based on probabilities.

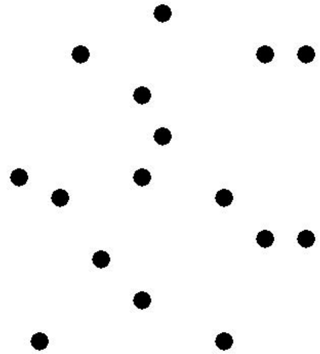
However, at least in vision, it requires

- a definition of visual regularity (to set the search space)
- computable simplest codes (to make actual predictions)

What is visual regularity?

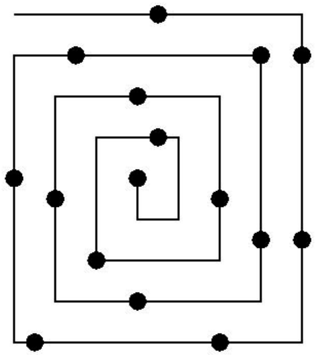
Symmetry and repetition would qualify as visual regularities

What is visual regularity?



Dots arranged equidistantly along an invisible rectangular spiral

What is visual regularity?

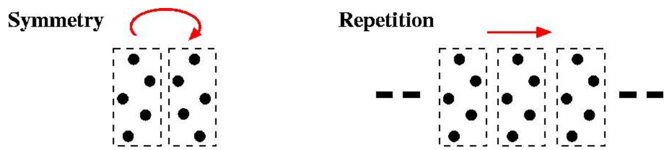


Dots arranged equidistantly along a visible rectangular spiral
– apparently, without the spiral, there is no a visual regularity

Visual regularity as form of invariance

Traditional approach (see Garner, 1974; Palmer, 1983):

- transformational regularity: invariance under motion



  regular patterns remain the same after rigid transformations
- relevant, but does not explain human regularity detection
- suited for object recognition but not for object perception

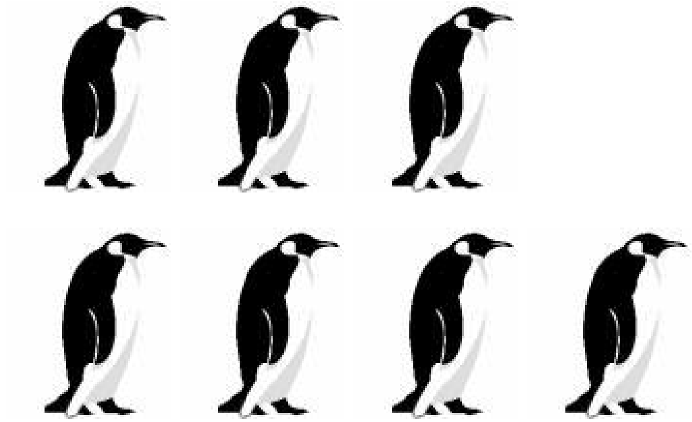
Alternative approach:

- holographic regularity: invariance under growth

Invariance under growth (informal)

Symmetry: structure that remains the same after (point-wise) growth.

Invariance under growth (informal)



Repetition: structure that remains the same after (block-wise) growth.

Formal step 1/7: Identity chains

• An identity $(i \quad i+1 \quad ... \quad i+k-1) = (j \quad j+1 \quad ... \quad j+k-1)$ or, for short, $I(i,j;k)$, in a string S is an identity relationship between two identical substrings $S_{i,k}$ and $S_{j,k}$ in S.

• An identity set $G = \{I(i_p, j_p; k_p) \mid p = 1, 2, ..., n; n \geq 1\}$ in a string S is an n-identity chain if:

1. $i_p + k_p \leq i_q$ for $p < q$,
2. $i_p + k_p \leq j_p$ for $p = 1, 2, ..., n$, and
3. the substrings S_{j_p,k_p} $(p = 1, 2, ..., n)$ are pairwise disjoint.

symbol string	*a b c p a b c*	*a b c p a b c*
identity chain	{(1)=(5), (2)=(6), (3)=(7)}	{(1 2)=(5 6), (3)=(7)}

Identity chains are ordered sets of identities.

Formal step 2/7: Complete chunking

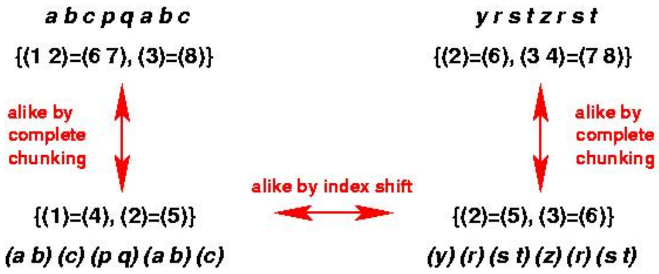
A complete chunking chunks a string in line with an identity chain that, in the resulting chunk string, transforms into its hierarchical image.

symbol string	*a b c p a b c*	*a b c p a b c*
identity chain	{(1 2 3)=(5 6 7)}	{(1 2)=(5 6), (3)=(7)}
complete chunking	(a b c) (p) (a b c)	(a b) (c) (p) (a b) (c)
hierarchical image	{(1)=(3)}	{(1)=(4), (2)=(5)}

An identity chain and its hierarchical image are "alike".

Formal step 3/7: Identity structures

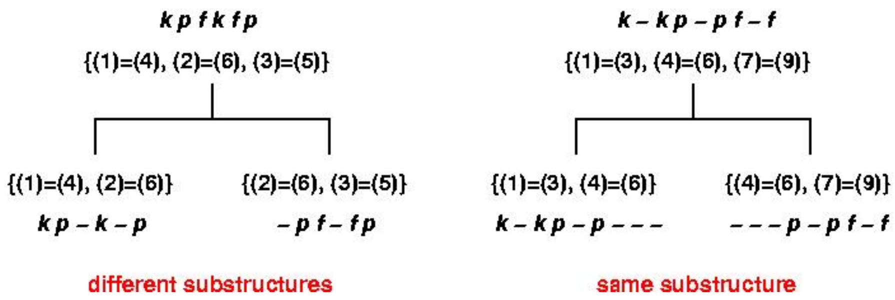
• For an n-identity chain G, the n-identity structure $\Sigma(G)$ is the set of all n-identity chains the hierarchical image of which is identical to the hierarchical image of G up to an index shift.
• $\Sigma(F)$ is an m-identity substructure of $\Sigma(G)$ if F is an m-identity subchain of G.



There are precisely 648 different 3-identity structures.

Formal step 4/7: The holographic property

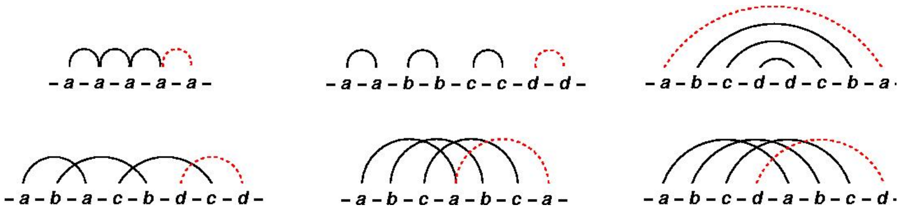
An n-identity structure is holographic if, for every fixed m ($1 \leq m < n$), all its m-identity substructures are the same.



Only 24 of the 648 3-identity structures are holographic, 20 of which can be expanded preserving the holographic property.

Formal step 5/7: Holographic regularities

A holographic regularity is a set $\{\Sigma(H_n) \mid n = 1, 2, ..., \infty\}$ of n-identity structures $\Sigma(H_n)$ with, for every $n \geq 2$ and every $m < n$: all m-identity substructures of $\Sigma(H_n)$ are identical to $\Sigma(H_m)$.

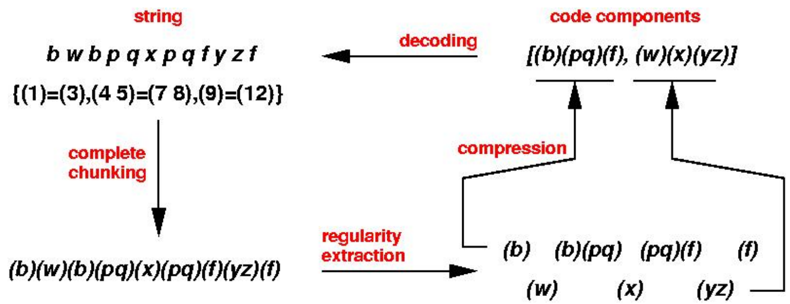


There are 6 holographic themes yielding, in total, 18 single and 2 classes of holographic regularities.

Formal step 6/7: Coding rules

A coding rule is a decoding function for a set of encodings.

For instance: $M[(k_1)...(k_n), (y_1)...(y_n)] \rightarrow k_1 y_1 k_1 \; k_2 y_2 k_2 \; ... \; k_n y_n k_n$



All instantiations of the holographic regularities can be captured by 80 syntactically coherent coding rules.

Formal step 7/7: Hierarchical transparency

A string code is hierarchically transparent if any regularity nested in it corresponds unambiguously to the same kind of regularity in the string.

S-rule: $S[(k_1)...(k_n),(y_1)] \rightarrow k_1k_2...k_n\ y_1\ k_nk_{n-1}\ ...\ k_1$
$byyfpfyyb \rightarrow S[(b)(y)(y)(f),(p)] \rightarrow S[(b)\ 2*((y))\ (f),(p)]$
$2*(y) \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow 2*((y))$
transparent

M-rule: $M[(k_1)...(k_n),(y_1)...(y_n)] \rightarrow k_1y_1k_1\ k_2y_2k_2\ ...\ k_ny_nk_n$
$bxbpypfyf \rightarrow M[(b)(p)(f),(x)(y)(y)] \rightarrow M[(b)(p)(f),(x)\ 2*((y))]$
?????? $\leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow 2*((y))$
not transparent

K-rule: $K[(k),(y_1)(y_2)...(y_n)] \rightarrow ky_1k\ ky_2k\ ...\ ky_nk$
$bxbbybbyb \rightarrow K[(b),(x)(y)(y)] \rightarrow K[(b),(x)\ 2*((y))]$
$2*(byb) \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow 2*((y))$
transparent

Accessibility as detection criterion

A holographically regular structure is made up of substructures expressing the same regularity, so, it is accessible by building on its substructures.

A transparent hierarchical structure is made up of nested structures that are also accessible separately.

Both accessibility features seem required for a regularity to qualify as a visual regularity.

Not all holographic regularities are transparent, nor vice versa:
- the regularity captured by the M-rule is holographic but not transparent
- the regularity captured by the K-rule is transparent but not holographic

A few regularities, however, are both holographic and transparent ...

Transparent holographic regularities

The formal criteria of holographic regularity and transparent hierarchy single out 3 regularities – these are proposed to be the visual regularities.

These 3 regularities are captured by the following coding rules:

- Iteration rule (I-rule):
  $n*(y) \rightarrow yyy...y$ (n times y)
- Symmetry rule (S-rule):
  $S[(k_1)...(k_n),(y_1)] \rightarrow k_1k_2...k_n\ y_1\ k_nk_{n-1}\ ...\ k_1$
- Alternation rule (A-rule):
  $\langle(y)\rangle/\langle(x_1)(x_2)...(x_n)\rangle \rightarrow yx_1\ yx_2\ ...\ yx_n$
  $\langle(x_1)(x_2)...(x_n)\rangle/\langle(y)\rangle \rightarrow x_1y\ x_2y\ ...\ x_ny$

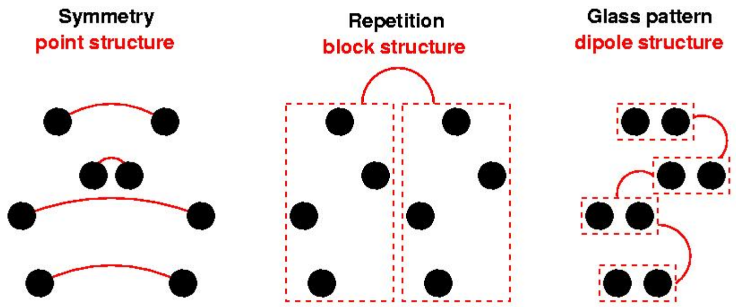
Motion or growth?

The traditional transformational approach focuses on properties of regular objects (originally in 3D) – the holographic approach goes deeper by focusing on properties of regularities (in 1D).

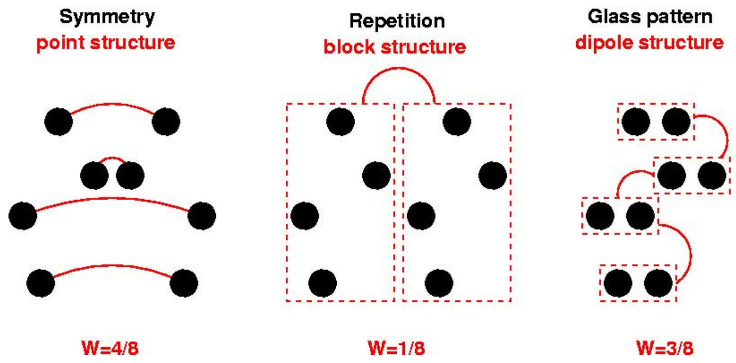
Yet, both approaches
- are mathematically elegant
- single out regularities that are perceptually relevant

What about human detection of visual regularity?
- not explained by the transformational approach
- explained by the holographic approach?

The holographic structure of visual regularities



Detectability: Holographic weight of evidence



If n is the number of elements in the pattern and E the number of holographic identities that form the regularity, then $W = E/n$ is the weight of evidence for the regularity in the pattern.

Empirical evidence

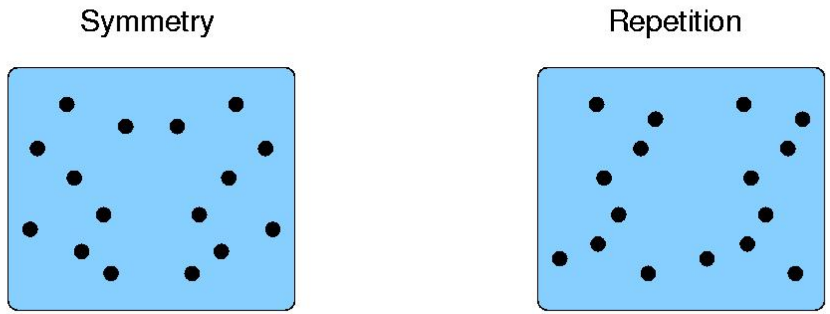
The holographic weight-of-evidence measure $W = E/n$ explains

- the basic phenomenon that symmetry and Glass patterns are about equally "good" and "better" than repetition

It further explains, among others:

- number effect in repetition but not symmetry (Csathó et al., 2003)
- graceful degradation of perturbed symmetry (Barlow & Reeves, 1979)
- symmetry and asymmetry effects (Csathó et al., 2004)

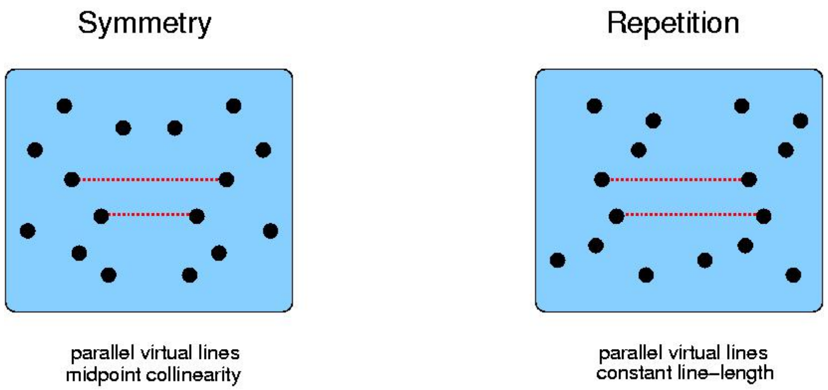
Holographic detection of visual regularity



Holographically, symmetry is "better" than repetition because

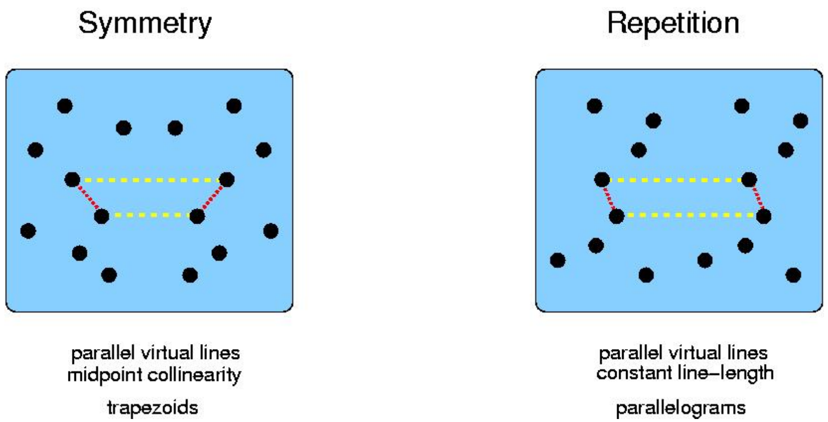
- symmetry detection propagates exponentially
- repetition detection propagates linearly

Detection anchors



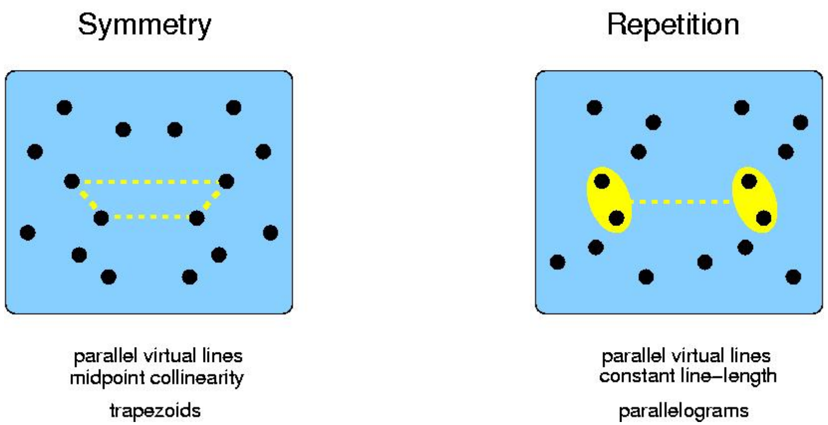
First-order anchors: virtual lines (Jenkins, 1983).

Detection anchors



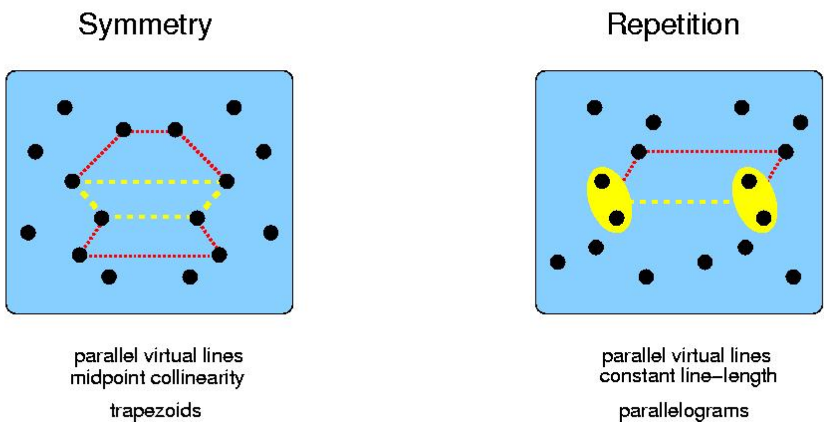
Second-order anchors: correlation quadrangles (Wagemans et al., 1993).

Holographic bootstrapping



Symmetry gets point structure, and repetition gets block structure.

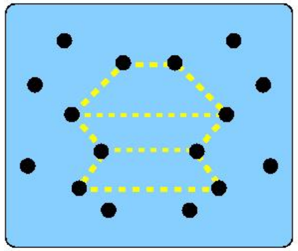
Holographic bootstrapping



Propagation: each virtual line forms an anchor for a new quadrangle ...

Holographic bootstrapping

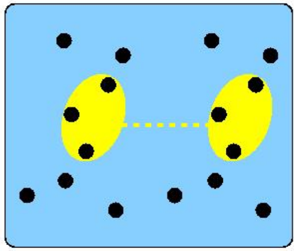
Symmetry



parallel virtual lines
midpoint collinearity

trapezoids

Repetition



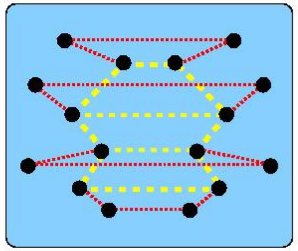
parallel virtual lines
constant line–length

parallelograms

... to add points in symmetry, and to enlarge the blocks in repetition.

Holographic bootstrapping

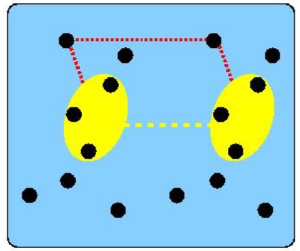
Symmetry



parallel virtual lines
midpoint collinearity

trapezoids

Repetition



parallel virtual lines
constant line–length

parallelograms

Thus, symmetry propagates exponentially, and repetition linearly.

Empirical evidence

Just as the holographic weight-of-evidence measure $W = E/n$, the holographic bootstrap mechanism explains:

- symmetry is "better" than repetition
- number effect in repetition but not symmetry

The holographic bootstrap mechanism explains further:

- heterogenities (eg. salient subpatterns) hinder symmetry detection but not repetition detection (Csathó et al., 2003)

  analogy: for a slow car (repetition) it hardly matters whether the road is busy, but for a fast car (symmetry) it does matter

Formalization: Conclusion

Just as the transformational approach, the holographic approach

- is mathematically elegant
- singles out regularities that are perceptually relevant

In addition, the holographic approach

- assigns a perceptually relevant structure to these regularities

Does it also make simplest descriptive codes computable?

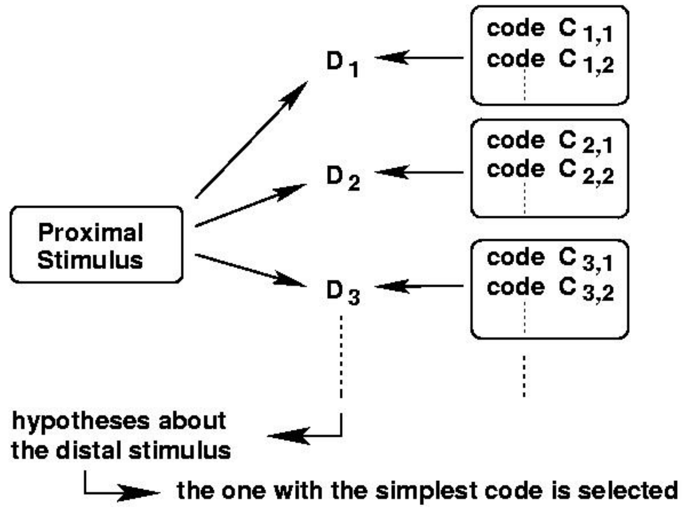
Structural information theory

Structural information theory (SIT) is a general theory of pattern perception, initiated by Emanuel Leeuwenberg in the 1960s and, since then, applied to a wide variety of perceptual phenomena.

It has been argued that SIT is the only approach to Gestalt psychology that spawned a formal calculus that can generate a plausible Gestalt representation of a perceptual phenomenon.

In line with the Gestalt law of Prägnanz, SIT applies the simplicity principle, that is, the preferred interpretation of a visual pattern is predicted to be the one with the simplest descriptive code.

SIT's simplicity principle



SIT's semantic mapping

SIT's formal calculus (ie. a coding language plus a complexity metric) is applied to primitive codes of hypothesized distal stimuli.

A primitive code is a symbol string that
(1) forms a reconstruction recipe in terms of perceptual primitives*,
(2) satisfies the spatial contiguity demand that juxtaposed symbols in the string refer to juxtaposed primitives in the hypothesized distal stimulus.

* no pre-fixed set of primitives; in the empirical practice, the primitives are chosen at the homogeneity level of a stimulus set, ie. just below the level of detail at which the stimuli start to show distinctive features.

With an eye at the Gestaltist hypothesis of psychophysical isomorphism, the idea then is that, in all its structural characteristics, SIT's encoding resembles psychological facts and underlying events in the brain.

SIT's coding language: transparent holographic rules

An SIT code $\overline{X}$ of a string X is a string $t_1 t_2 ... t_m$ such that $X = D(t_1)...D(t_m)$, where the decoding function $D : t \rightarrow D(t)$ takes one of the following forms:

I-form:	$n * (\overline{y})$	$\rightarrow$	$yyy...y$ (n times y)
S-form:	$S[\overline{(\overline{x_1})(\overline{x_2})...(\overline{x_n})}, (\overline{p})]$	$\rightarrow$	$x_1 x_2 ... x_n \; p \; x_n ... x_2 x_1$
A-form:	$\langle(\overline{y})\rangle / \langle\overline{(\overline{x_1})(\overline{x_2})...(\overline{x_n})}\rangle$	$\rightarrow$	$yx_1 \; yx_2 \; ... \; yx_n$
A-form:	$\langle\overline{(\overline{x_1})(\overline{x_2})...(\overline{x_n})}\rangle / \langle(\overline{y})\rangle$	$\rightarrow$	$x_1 y \; x_2 y \; ... \; x_n y$

Otherwise: $D(t) = t$

SIT codes (examples for one and the same string)

String: $X = ab\ acdacd\ ababa\ cdacda\ b$
Code 1: $\overline{X} = ab\ \ 2*(acd)\ \ S[(a)(b),(a)]\ \ 2*(cda)\ b$

String: $X = ab\ acd\ acd\ ab\ ab\ acd\ acd\ ab$
Code 2a: $\overline{X} = S[(ab)(acd)(acd)(ab)]$
Code 2b: $\overline{X} = S[\ S[((ab))((acd))]\ ]$

String: $X = ab\ acd\ acd\ ab\ \ ab\ acd\ acd\ ab$
Code 3a: $\overline{X} = 2*(ab\ acd\ acd\ ab)$
Code 3b: $\overline{X} = 2*(\langle(a)\rangle/\langle(b)(cd)(cd)(b)\rangle)$
Code 3c: $\overline{X} = 2*(\langle(a)\rangle/\langle S[((b))((cd))]\rangle)$

SIT's complexity measure: no bits but sips

Because SIT's coding rules are hierarchically transparent, a SIT code of a string describes a hierarchical organization of the string:

aaaa	*aaaa*	*abababab*	*abababab*
$4*(a)$	$2*(2*(a))$	$4*(ab)$	$2*(2*(ab))$
$(a)(a)(a)(a)$	$((a)(a))((a)(a))$	$(ab)(ab)(ab)(ab)$	$((ab)(ab))((ab)(ab))$
$I = 1$ sip	$I = 2$ sip	$I = 3$ sip	$I = 4$ sip

The complexity, or structural information load I, of a code then is given by the number of structural information parameters (sip) in the code, ie. by the number of different constituents (primitives and groups of primitives) in the hierarchical organization described by the code.

SIT's complexity measure: a few specifics

Further clarifications (ie. not further constraints):

(1) If the code does not describe (groups of) primitives as being identical, then they are counted as if they were different.

(2) Higher-level regularity in an S-form applies to one symmetry half.

(3) In an A-form, the repeat may seem separated from the rest but, in the hierarchical organization, it is still intertwined with the rest.

aaaa	*ababbaba*	*abacacab*
$2*(aa)$	$S[2*((a)(b))]$	$\langle(a)\rangle/\langle S[((b))((c))]\rangle$
$(aa)(aa)$	$((a)(b))((a)(b))(b)(a)(b)(a)$	$((a)(b))((a)(c))((a)(c))((a)(b))$
$I = 3$ sip	$I = 3$ sip	$I = 5$ sip

Note: SIT's complexity metric might need further fine-tuning but, as it stands, it is theoretically consistent and, thus far, it proved to be empirically valid.

Are simplest SIT codes computable?

A string of length N can be encoded in a super-exponential number of ways, ie. it has $O(2^{N \log N})$ different SIT codes.

String:	$X = abacdacdababacdacdab$	20 sip
Code 1:	$\overline{X} = ab\ \ 2*(acd)\ \ S[(a)(b),(a)]\ \ 2*(cda)\ b$	14 sip
Code 2a:	$\overline{X} = S[(ab)(acd)(acd)(ab)]$	14 sip
Code 2b:	$\overline{X} = S[\ S[((ab))((acd))]\ ]$	7 sip
Code 3a:	$\overline{X} = 2*(ab\ acd\ acd\ ab)$	11 sip
Code 3b:	$\overline{X} = 2*(\langle(a)\rangle/\langle(b)(cd)(cd)(b)\rangle)$	10 sip
Code 3c:	$\overline{X} = 2*(\langle(a)\rangle/\langle S[((b))((cd))]\rangle)$	8 sip

etc., etc., ...

Simplest codes may be computable in theory, but do not seem computable in practice – at least, not at first glance.

Which way out?

Easy ways out:

- "We model systematicity in the outcomes, not in the process"
  - Valid temporary stance, but not satisfactory in the long run
- "We use heuristics, and settle for suboptimal outcomes"
  - Might simulate human data, but is begging the question

The hard way out:

- "We adopt the simplicity principle as basic process law, and study suboptimal outcomes in humans to infer further laws"
  - But then optimal outcomes are supposed to be computable, which seems impossible under traditional forms of processing ...

Basic forms of processing

Traditional dichotomy:

- Serial processing:
  items are processed one after the other by one processor
- Parallel processing:
  items are processed simultaneously by many processors

In practice, processes are often composed of both serial and (potentially) parallel subprocesses.

Compared to serial processing, parallel processing reduces the amount of time to complete a job but not the amount of work.

Distributed processing

Distributed processing is not so much a form of processing but refers primarily to the fact that the items to be processed are stored in a distributed representation that also stores relationships between the items.

In cognitive science, mental representations are nowadays often assumed to be distributed representations.

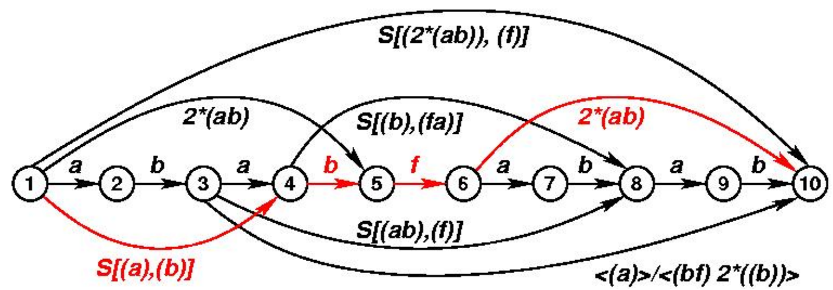
For instance, a distributed representation may implement the fact that items have parts in common (like routes in a road map).

A "smart" process may effectively use this fact to reduce the amount of work to complete a job and, thereby, also the amount of time.

This method is, since the 1950s, common property in computer science and stands apart from the issue of serial and/or parallel processing.

Distributed representations of SIT codes

Assume (just for the moment) that, for the string *ababfabab*, a simplest ISA-form is known for each of the $O(N^2)$ substrings (a few are shown):

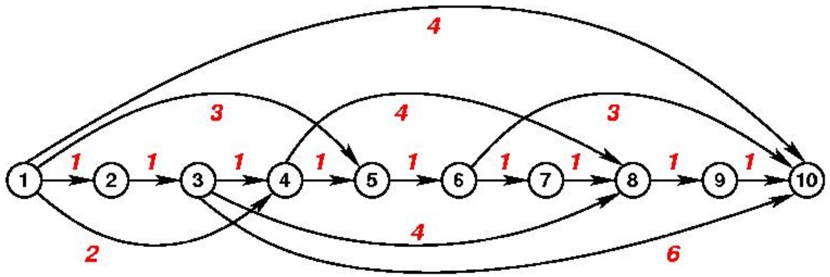


Then, $O(2^N)$ codes for the entire string are possible – for instance, the path along nodes 1, 4, 5, 6, 10 yields code $S[(a),(b)]\ b\ f\ 2*(ab)$

How to select the simplest code?

Serial distributed processing to select simplest codes

Take the complexities of the ISA-forms as the lengths of the links ...



and apply Dijkstra's (1959) shortest path method which yields the minimal distance $d_{min}(1, N)$ from node 1 to node N, by determining

$$d_{min}(1, k) = \text{MIN}_{p<k}\{d_{min}(1, p) + d(p, k)\} \qquad \text{for } k = 2, 3, ..., N.$$

This is a "smart" $O(N^2)$ method to evaluate $O(2^N)$ paths.

Serial versus parallel distributed processing

So-called "neural" network models describe cognitive functions in terms of activation spreading through distributed representations, which reflects parallel distributed processing.

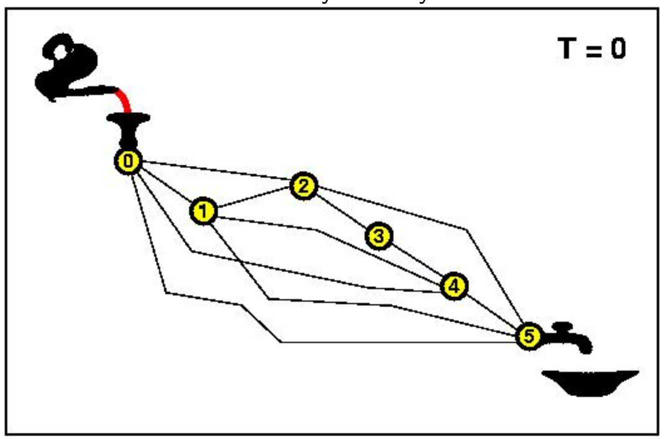
These network models, however, rarely know hardware implementations but are usually simulated on a computer by software implementations.

This does not affect the underlying processing ideas, but the software implementations actually perform serial distributed processing.

Inversely, many "smart" algorithms performing serial distributed processing (like Dijkstra's shortest path method) could be given a hardware implementation performing parallel distributed processing ...

Parallel distributed processing to select shortest paths

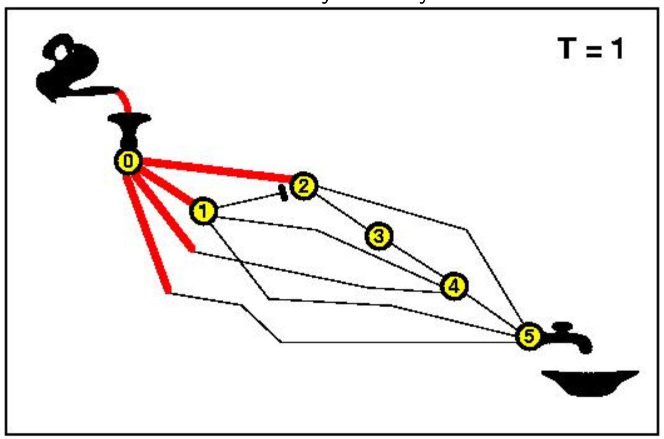
"Smart" hilly tube system



Fluid takes one time unit to "exite" one straight tube segment

Parallel distributed processing to select shortest paths

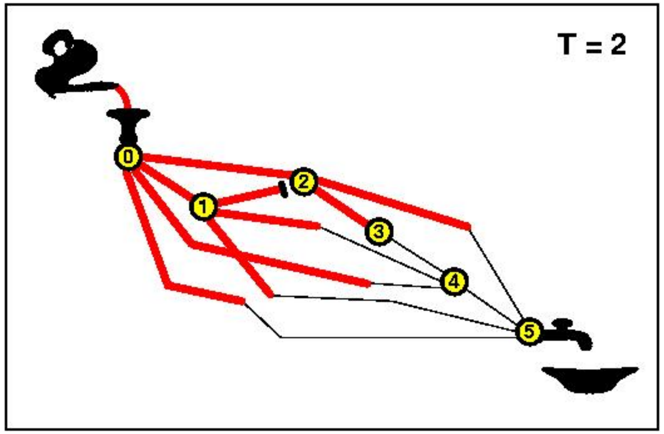
"Smart" hilly tube system



Fluid, arriving at a node, "inhibits" other incoming tubes

Parallel distributed processing to select shortest paths

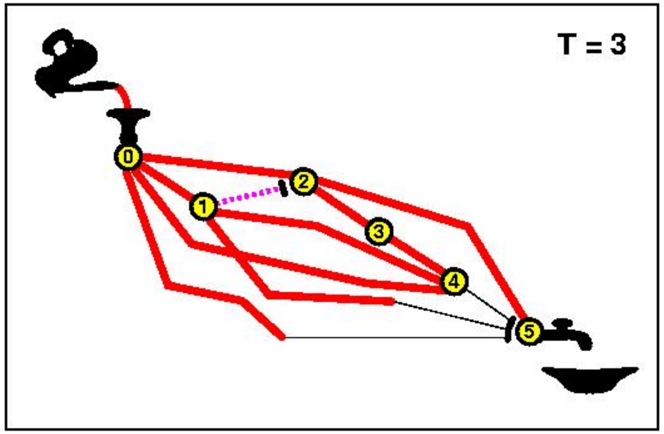
"Smart" hilly tube system



Fluid flows on, also in blocked tubes

Parallel distributed processing to select shortest paths

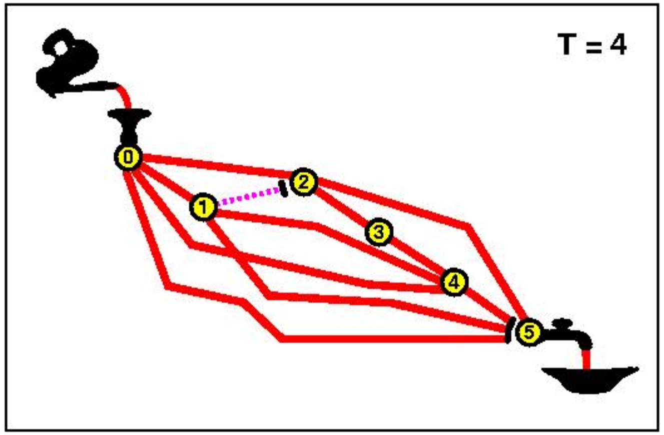
"Smart" hilly tube system



Fluid in blocked tubes hardens in one time unit

Parallel distributed processing to select shortest paths

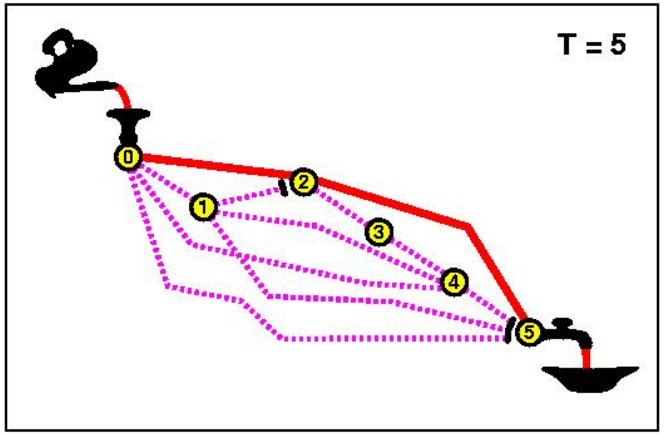
"Smart" hilly tube system



Fluid exits through the shortest path

Parallel distributed processing to select shortest paths

"Smart" hilly tube system



All other paths harden, leaving a flow in only the shortest path

SIT versus "neural" network models

- SIT's encoding of input starts with the (potentially parallel) detection of local structures[1] which are stored in a distributed representation[2].

[1] not by testing all relevant structures that occur in the set of all possible inputs, but by applying a few rules that extract only the relevant structures in any input at hand (storage friendly).
[2] not prefixed but constructed on the fly, with nodes impersonating spatial relationships between structures stored in the links (an activation flow does not activate information but is information).

- SIT's encoding results in a global structure, impersonated by a trace, and selected by way of (potentially parallel) distributed processing.

Hence, SIT does not assume one network for all inputs, but yields input dependent networks; for the rest, the two subprocesses above do not differ essentially from processes put forward in "neural" network models.

However, SIT's encoding also involves hierarchical encoding ...

The problem of SIT's hierarchical encoding

Before, we assumed a simplest ISA-form was known for each substring.

However, to find a simplest ISA-form for a substring, every S-form and A-form for the substring has to be encoded hierarchically:

$$ababgababbabagabab$$

$$2*(ab)\ gababbabag\ 2*(ab) \qquad ababg\ S[(a),(b)]\ bb\ S[(a),(b)]\ gabab$$
$$\rightarrow S[(2*(ab))\ (g)(a)(b)(a)(b)] \qquad \rightarrow S[(ab)(ab)(g)\ (S[(a),(b)])\ (b)]$$
$$\rightarrow S[(2*(ab))\ (g)\ 2*((a)(b))] \qquad \rightarrow S[2*((ab))\ (g)\ (S[(a),(b)])\ (b)]$$

A substring of length k has $O(2^k)$ S-forms and $O(k2^k)$ A-forms, each of which has to be encoded hierarchically, with $O(\log k)$ recursion steps.

Traditional forms of processing cannot do this realistically.

Are there other forms of processing?

	One item at a time	Many items at a time
One processor	Serial processing	?
Many processors	?	Parallel processing

Other forms of processing

	One item at a time	Many items at a time
One processor	Serial processing	?
Many processors	Subserial processing	Parallel processing

Think of supermarket customers (the "processors") who, one after the other, empty their carts (the "items") at the checkout.

Going beyond parallel processing

	One item at a time	Many items at a time
One processor	Serial processing	Transparallel processing
Many processors	Subserial processing	Parallel processing

This may seem science-fiction, but ...

Selection of longest pencil (1)

Measure the pencil lengths serially or in parallel ..... not smart!

Selection of longest pencil (2)

Is this serial or parallel processing? No, it is transparallel processing!

No science-fiction

The foregoing shows that transparallel processing (many items at a time by one processor)

- is not science-fiction
- reduces both work and time needed to complete a job

Distributed representations also reduce both work and time, so, the combination would give a double reduction!

Again, no science-fiction ...

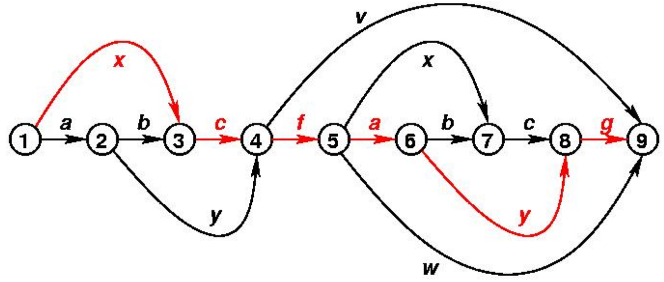
Hyperstrings

Strings	x c v	a b c f w	a y f x c g
	a b c f a b c g	a b c f a y g	a y f a y g
	a b c v	x c f w	a y f a b c g
	x c f a b c g	x c f x c g	a b c f x c g
	a y v	x c f a y g	a y f w

Hyperstrings

Strings	x c v	a b c f w	a y f x c g
	a b c f a b c g	a b c f a y g	a y f a y g
	a b c v	x c f w	a y f a b c g
	x c f a b c g	x c f x c g	a b c f x c g
	a y v	x c f a y g	a y f w

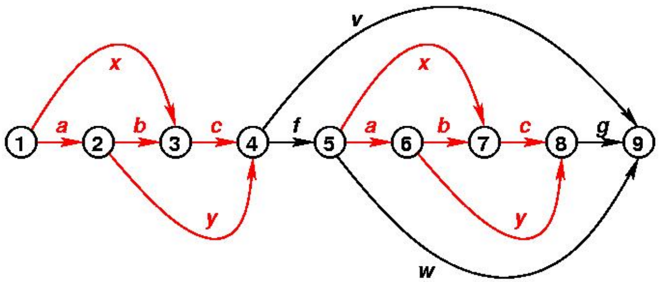
Distributed representation



Hyperstrings

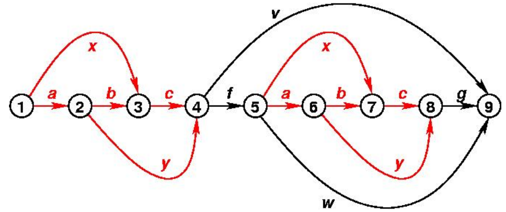
Strings	x c v	a b c f w	a y f x c g
	a b c f a b c g	a b c f a y g	a y f a y g
	a b c v	x c f w	a y f a b c g
	x c f a b c g	x c f x c g	a b c f x c g
	a y v	x c f a y g	a y f w

Distributed representation



Hyperstring: hypersubstrings are 0% or 100% identical – never something in between

Encoding a hyperstring



Here, the hypersubstrings $h_{1,3}$ and $h_{5,3}$ are identical, so, this hyperstring can be encoded into, for instance, the A-form $\langle(h_{1,3})\rangle/\langle(h_{4,1})(h_{8,1})\rangle$

This A-form represents, in one go, the A-forms
- $\langle(abc)\rangle/\langle(f)(g)\rangle$ in the string *abcfabcg*
- $\langle(xc)\rangle/\langle(f)(g)\rangle$ in the string *xcfxcg*
- $\langle(ay)\rangle/\langle(f)(g)\rangle$ in the string *ayfayg*

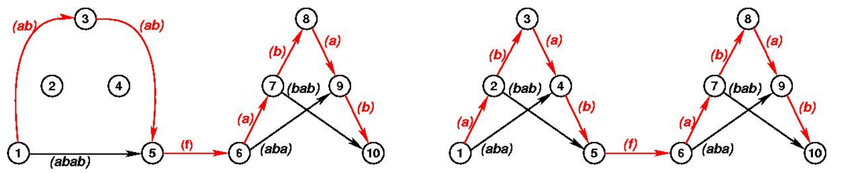
Hence, by encoding a hyperstring, all represented strings are encoded in one go, that is, in a transparallel fashion.

SIT's hierarchical encoding for a substring of length k

Gather the $O(2^k)$ S-arguments in a distributed representation:

ababfababgbabafabab
$S[(ab)(ab)(f)(a)(b)(a)(b),(g)]$

ababfababgbabafbaba
$S[(a)(b)(a)(b)(f)(a)(b)(a)(b),(g)]$



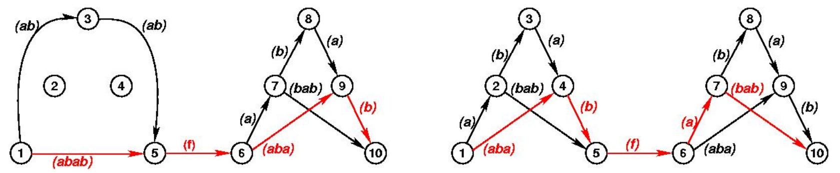
Relevance | Structural information theory
Formalization | Forms of (cognitive) processing
Processing | Transparallel processing by hyperstrings

SIT's hierarchical encoding for a substring of length k

Gather the $O(2^k)$ S-arguments in a distributed representation:

ababfababgbabafabab
$S[(ab)(ab)(f)(a)(b)(a)(b),(g)]$
$S[(abab)(f)(aba)(b),(g)]$

ababfababgbabafbaba
$S[(a)(b)(a)(b)(f)(a)(b)(a)(b),(g)]$
$S[(aba)(b)(f)(a)(bab),(g)]$



Peter A. van der Helm | Visual Regularity

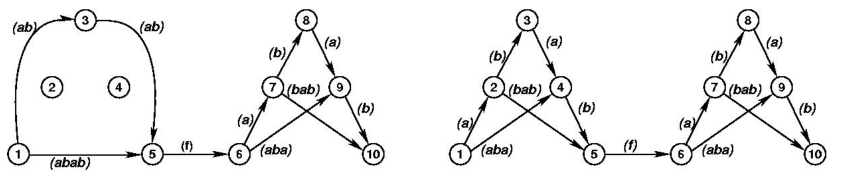
Relevance | Structural information theory
Formalization | Forms of (cognitive) processing
Processing | Transparallel processing by hyperstrings

SIT's hierarchical encoding for a substring of length k

Gather the $O(2^k)$ S-arguments in a distributed representation:

ababfababgbabafabab
$S[(ab)(ab)(f)(a)(b)(a)(b),(g)]$
$S[(abab)(f)(aba)(b),(g)]$
etc., etc., ...

ababfababgbabafbaba
$S[(a)(b)(a)(b)(f)(a)(b)(a)(b),(g)]$
$S[(aba)(b)(f)(a)(bab),(g)]$
etc., etc., ...



Such a representation can be built in $O(k^2)$ steps and is, by nature, a hyperstring, so, it enables transparallel encoding of the S-arguments.

This has been implemented (also for A-arguments) in the algorithm PISA which computes a guaranteed simplest SIT code for any symbol string.

Peter A. van der Helm | Visual Regularity

Relevance | Structural information theory
Formalization | Forms of (cognitive) processing
Processing | Transparallel processing by hyperstrings

Processing: Conclusion

Compared to nondistributed representations, distributed representations may reduce both work and time needed to process typically $O(2^N)$ items:

- both work and time may be reduced typically from $O(2^N)$ to $O(N^2)$.

Compared to serial processing, parallel processing requires more processors, and reduces time but not work needed to complete a job:

- applied to distributed representations, the total reduction in time is typically from $O(2^N)$ to $O(N)$.

Transparallel processing does not require more processors, but does reduce both work and time needed to complete a job:

- applied to distributed representations called hyperstrings, the total reduction in both work and time is typically from $O(2^N)$ to $O(1)$.

SIT's transparent holographic regularities can be encoded hierarchically by way of transparallel processing applied to hyperstrings.

Peter A. van der Helm | Visual Regularity

Summary

- A general-purpose predictive system based on simplest descriptive codes is a good alternative to special-purpose predictive systems based on often not quantifiable probabilities, if it provides
  - a definition of visual regularity, and
  - computable simplest descriptive codes.
- The formal notion of transparent holographic regularity
  - singles out regularities that are perceptually relevant, and
  - assigns a perceptually relevant structure to these regularities.
- Transparallel processing (many items at a time by one processor)
  - makes simplest descriptive codes computable, and
  - might well occur in a brain attuned to regularities in the world.

Relevance
Formalization
Processing

Further reading

- Relevance
  - van der Helm, P. A. (2000). Simplicity versus likelihood in visual perception: From surprisals to precisals. *Psychological Bulletin, 126*, 770–800.
- Formalization
  - van der Helm, P. A., & Leeuwenberg, E. L. J. (1991). Accessibility, a criterion for regularity and hierarchy in visual pattern codes. *Journal of Mathematical Psychology, 35*, 151–213.
  - van der Helm, P. A., & Leeuwenberg, E. L. J. (1996). Goodness of visual regularities: A nontransformational approach. *Psychological Review, 103*, 429–456.
- Processing
  - van der Helm, P. A. (2004). Transparallel processing by hyperstrings. *Proceedings of the National Academy of Sciences USA, 101 (30)*, 10862–10867.

Peter A. van der Helm Visual Regularity