ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Iterative Context Compilation for Visual Object Recognition

Jens Teichert and Rainer Malaka

European Media Laboratory GmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
jens.teichert@eml.org

Abstract. This contribution describes an almost parameterless iterative context compilation method, which produces feature layers, that are especially suited for mixed bottom-up top-down association architectures. The context model is simple and enables fast calculation. Resulting structures are invariant to position, scale and rotation of input patterns.

1 Introduction

Images are influenced by change of illumination, perspective transformation or varying subcomponents of scene objects. Therefore vision architectures have to extract something from the original feature layer (i.e. image pixels) and create higher (more abstract) invariant representations. Classical vision architectures usually create higher feature layers by weighting and summing up intensity values of neighboring pixels in the input image. The contribution of the investigated neighborhood is given by a rigid mask. Position invariance can be accomplished, if masks and weights are equal (shared) for each pixel. Some invariance can be gained, if layers are stacked and sub-sampled iteratively [1, 2]. Additionally, scale invariance is achievable, if features are detected at different resolutions [3]. Higher variances are yet not manageable with rigid masks [4]. A major problem with this kind of layered architectures is, that they have to deploy discrete masks for continuous variance characteristics, e.g. scale or rotation. This results in different output layers with generalization problems for following associations.

Other approaches calculate context descriptors, which are invariant under perspective transformation, e.g. [5, 6]. But this is problematic, when objects appear with varying subcomponents like most everyday objects. The only known solution to that problem is a hierarchical processing of invariant subcomponents, which are efficient and can be combined to describe an object [7, 8]. The method proposed in this paper is part of an architecture, which tries to implement such a hierarchy. The processing starts from pixel level and detects feature layers with increasing abstraction of representation. The processing on the feature layers is described in this paper.

2 Iterative Context Compilation

The input representation of our iterative context compilation method consists of two scalar values for each pixel. The scalar v represents the value of a feature,

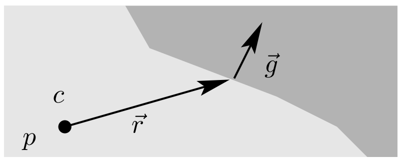


Fig. 1: Context Model. Values v are given by grey values on the feature layer. For a position p on the layer, context information is given by the most essential gradient $\vec{g}$ of the environment, its translation vector $\vec{r}$ and their confidence c.

which is represented by the layer. On the lowest layer, this is the brightness of a pixel. On higher layers, v can represent more spacious and abstract features like the the curvature of a line or the skewness of a rectangle, etc. The scalar c^v describes the confidence of the value v. For the lowest (input image) layer the confidence is set to one. On higher layers the confidence denotes the quality of the association, that produced v.

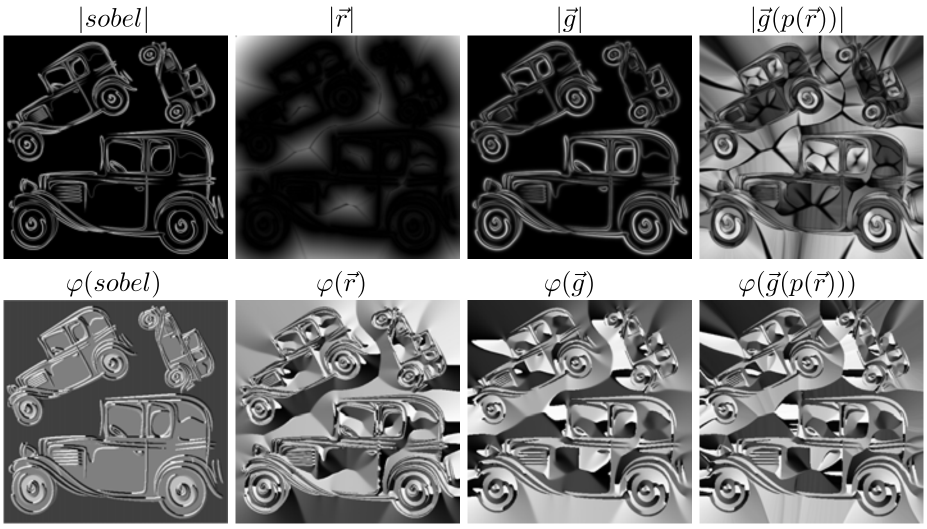
From v and c^v, the context compilation calculates the dominating local gradient $\vec{g}$, its translation vector $\vec{r}$ and their confidence c (Fig. 1). This model can describe simple spatial arrangements on a layer. More complex and abstract representations can evolve on following higher layers using associations, which combine underlying arrangements and produce new feature layers as output (this is not subject of this paper).

In contrast to classical mask orientated feature extraction, our method also describes image structures in areas, where no input activation is given (Fig. 2). This is achieved using an iterative information forwarding, which provides every pixel with context information. Moreover, this approach can be extended for top-down changes of representations during the recognition process in parallel with the context iteration, which is essential for visual pattern recognition [9].

Each feature layer is divided into cells ordered on a regular grid. We only consider equilateral triangles, squares or hexagons. Every cell is indexed by a position p, which might be a simple tuple (i, j) for the classical squares grid. Values of neighbors of p are indexed by a lowered character. Sums over a neighbor index include all elements of a dedicated neighborhood type, which can involve different orders of neighborhoods. Table 1 shows orders, the amount of participants and a corresponding geometric distance factor σ.

	border	1^{st} Neighbors		2^{nd} Neighbors		3^{rd} Neighbors	
topology	length	#	σ	#	σ	#	σ
triangles	$\sqrt{3}$	3	1	6	$\sqrt{3}$	3	2
squares	1	4	1	4	$\sqrt{2}$	-	-
hexagons	$1/\sqrt{3}$	6	1	-	-	-	-

Table 1: Neighborhood distance factors σ in dependence of cell topologies, given cell gravity distances normalized to one. Values are shown for neighbors with direct contact to the center cell only.



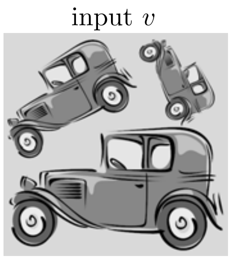


Fig. 2: A classical mask operator (e.g. sobel) fills areas with input activity within mask range only. The proposed method fills the the complete feature layer with context information of the next dominant gradient $\vec{g}$ of v. Context is described by the translation vector $\vec{r}$ and the gradient $\vec{g}$, which can be referenced by the translated position $p(\vec{r})$. Note that the images show some artefacts due to the carry of the angle projection.

At every iteration at time step t we first calculate local variables for each cell and its first adjacent neighbors which are indexed by b. These variables are the averaged local confidence c^l and the local gradient $\vec{g}^{\,l}$ which point in the direction $\vec{e}_b$ of the neighbor cell. Figure 3 shows the possible structure of the representation and locations for a squared neighborhood grid.

$$c^l_b(p,t) \;=\; \frac{c^v(p,t)+c^v_b(p,t)}{2}\,, \quad \vec{g}^{\,l}_b(p,t) \;=\; \big(v_b(p,t)-v(p,t)\big)\,\vec{e}_b$$

In the second step, for each cell three variables are calculated, that are used for an iterative information exchange between cells. The neighborhood cells are indexed with a and use first and second neighbors in our simulations. The gradient $\vec{g}(p,t)$ shows the dominant change of v in the environment of position p at time step t and the translation vector $\vec{r}(p,t)$ where this change occurs relative to the position p. The confidence $c(p,t)$ reflects the reliability of the two vectors.

The confidence for the next time step sums up and weights local and exchange confidence values:

$$c(p,t+1) \;=\; \frac{1}{\mu}\big(\lambda\sum_b \sigma_b\, c^l_b(p,t)+\sum_a \sigma_a\, c_a(p,t)\big); \;\; \mu \;=\; \lambda\sum_b \sigma_b+\sum_a \sigma_a$$

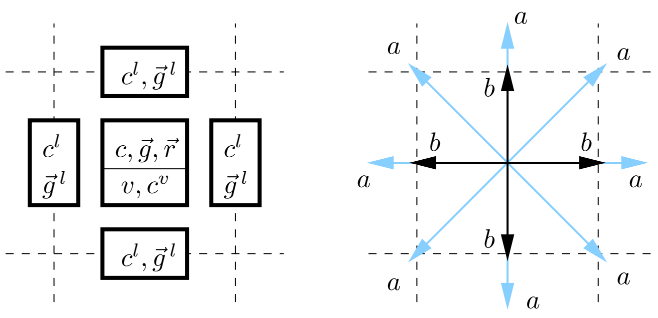


Fig. 3: Example of a cell in a squared grid with cell boundaries with first neighbors (b) for local values c^l and $\vec{g}^l$, first and second neighbors (a) for iteration exchange of c, $\vec{g}$ and $\vec{r}$ between cells.

The factor μ normalizes the geometric distances and a factor λ weights the local values. The gradient for the next time step is calculated by summation and weighting neighbor gradients according to the neighbor confidence:

$$\vec{g}(p,t+1) \quad = \quad \frac{1}{\mu}\big(\lambda \sum_b \frac{\sigma_b\, c_b^l(p,t)}{c(p,t+1)}\, \vec{g}_b^{\,l}(p,t) + \sum_a \frac{\sigma_a\, c_a(p,t)}{c(p,t+1)}\, \vec{g}_a(p,t)\big)$$

The translation vector for the next time step is calculated by the sum of the translation vectors in the neighborhood $\vec{r}_a(p,t)$ increased by the vectors $\vec{e}_a$ in the direction of the neighbor. Additionally, the results are weighted be the corresponding gradient contributions $\vec{g}_a(p,t)$. A simple proportion would by the length of projection on the new gradient. But this yields very large values, if some neighbors have gradients in opposite direction, while others are near to zero. Therefore we use a proportion with similar results but without these outliers, by taking absolute terms only:

$$\vec{r}(p,t+1) \quad = \quad \frac{\sum_a \Big(\sigma_a\, c_a(p,t)\ \big|\big\langle \vec{g}_a(p,t), \vec{g}(p,t+1)\big\rangle\big|\ \left(\vec{r}_a(p,t) + \vec{e}_a\right)\Big)}{\sum_a \Big(\sigma_a\, c_a(p,t)\ \big|\big\langle \vec{g}_a(p,t), \vec{g}(p,t+1)\big\rangle\big|\Big)}$$

Note, that there are no contributions from boundary gradients $\vec{g}_b^{\,l}$. But they are included in the new gradient $\vec{g}(p,t+1)$ and thereby decrease the new translation vector by a zero contribution indirectly, i.e., a local activation decrease the translation vector.

3 Experimental Results and Conclusions

Simulations shown in Fig. 2 and 4 are calculated with $\lambda = 0.1$ and 200 iterations. They show, that the context compilation method yields similar structures, if patterns are changed in position, scale or rotation and they show, that the position referencing of $\vec{g}$ works properly.

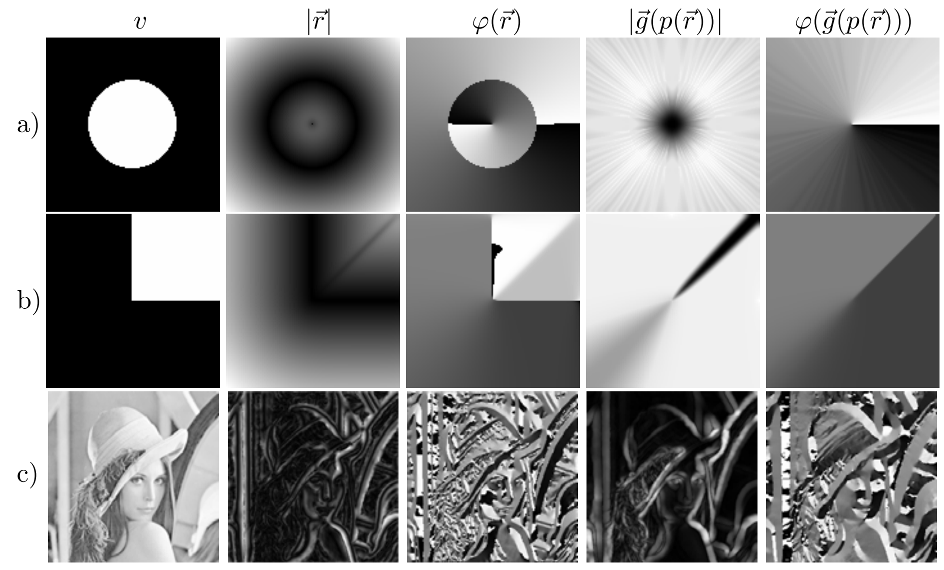


Fig. 4: Experimental results for technical and natural patterns (128x128 pixel) after 200 iterations (10ms/iteration on a current PC).

The method is parameterized by λ only. If it is set too high, the iteration will get stuck somewhere within the layer. Otherwise the method yields similar results over a wide range of λ and converges always.

The feature layer is filled completely by context information, which is advantageous for association architectures, that should detect spacious distributed structure. The context compilation is very efficient for detection, if the representation on higher layers is getting more sparse, because context information can be transported over wide areas. In contrast, mask functions are computationally very expensive, if applied in groups with different scalings or rotations over wide areas.

A confidence calculation is integrated to facilitate mixed bottom-up and top down processing of representation probabilities. Fig. 5 shows that the method can cope with iterative changes of representations, which is important, when activations are suppressed by top-down disambiguation. The layer may not be recalculated completely for an input change. It can adapt itself in parallel to input changes over time.

The method allows for very easy resolving of scale or rotation transformations by top-down scaling or rotating $\vec{r}$ and $\vec{g}(p(\vec{r}))$ globally or even regionally.

References

[1] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

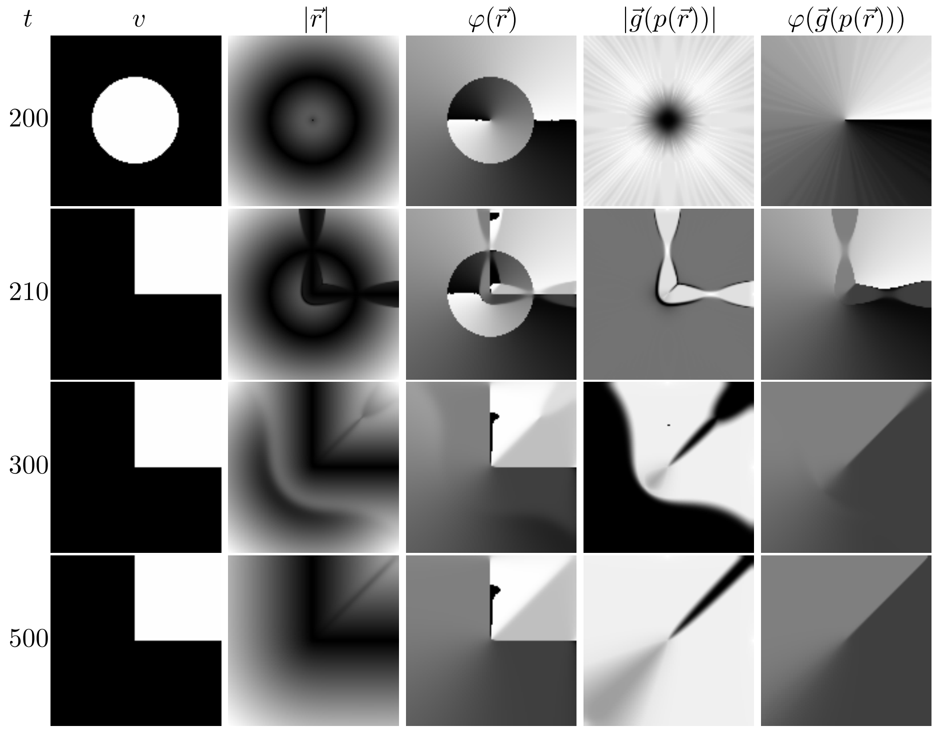


Fig. 5: Iterations after an input change at $t = 200$. The context information changes accordingly over the following iterations.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[3] J. Teichert and R. Malaka. An association architecture for the detection of objects with changing topologies. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2003*, pages 125–130, Portland, USA, 2003.

[4] Laurenz Wiskott. How does our visual system achieve shift and size invariance. In J. L. van Hemmen and T. J. Sejnowski, editors, *Problems in Systems Neuroscience.* Oxford University Press, 2004.

[5] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation by image exploration. In T. Pajdla and J. Matas, editors, *European Conference on Computer Vision (ECCV)*, volume 1, pages 40–54. Springer, May 2004.

[6] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *Journal of Computer Vision*, 60(1):63–86, 2004.

[7] W. A. Wickelgren. Context sensitive coding, associative memory and serial order in (speech) behavior. *Psychological Review*, 76:1–15, 1969.

[8] B. W. Mel and J. Fiser. Minimizing binding errors using learned conjunctive features. *Neural Computation*, 12:731–762, 2000.

[9] Bruno A. Olshausen and David J. Field. How close are we to understanding V1. *Neural Computation*, 17:1665–1699, 2005.