Cognitive object recognition based on dynamic field theory

Oliver Lomp

October 18, 2016

Cognitive object recognition based on dynamic field theory

(Kognitive Objekterkennung auf Basis der dynamischen Feldtheorie)

Oliver Lomp
Geboren in Recklinghausen

Bochum, Oktober 2016

Dissertation zur Erlangung des Grades eines Doktor-Ingenieurs der
Fakultät für Elektrotechnik und Informationstechnik an der
Ruhr-Universität Bochum.

Abstract

Our brains excel at parsing the vast amount of visual information that enters through our eyes. Replicating this level of skill in artificial vision systems has proven difficult. To date, the most successful approaches mimic the fast, feedforward categorization capabilities of our visual system, but its other important abilities, for example, quickly learning to recognize new objects in previously unseen orientations have not yet been modeled as successfully. In part, this is due to a gap in our understanding of the functional architecture of the visual system. One model that aims to reduce this gap uses localized feature histograms to quickly learn to recognize objects while concurrently estimating their pose. The system is capable of recognizing objects in new poses based on just a small number of training views using principles of dynamic field theory, a mathematical framework for modeling cognitive processes. I extend this object recognition model by the capacity to sequentially attend and recognize items in a scene, and the capacity to form working memory for the results of these recognitions. This extension is based on integrating the object recognition system with a model for scene representation also based on dynamic field theory. As part of this integration, I enable the object recognition system to recognize multiple objects in succession based only on principles of neural dynamics. The result is a large dynamical system that autonomously recognizes objects in a scene, fully implemented within the framework of dynamic field theory. I assess this system on a new database that contains multiple objects, on which the system achieves acceptable performance.

I present a second extension of the object recognition system, which enables an object representation that preserves the spatial arrangement of feature values better than the previously used localized histograms. Using this new representation further grounds the object recognition system in neural principles. It also allows the system to estimate scale and the full range of object orientations. In addition, it uncovers some limitations of the pose estimation and recognition process, which I address by introducing a modified dynamics for representing the recognition result. I demonstrate the performance of the system, showing that recognition rates improve with the new representation at comparable sampling rates. I also compare the system's performance to behavioral data on human object recognition and find that this view-based object recognition system exhibits pose-invariant performance, which contrasts with opinions in the literature that interpret evidence of pose-invariant performance as a sign that object descriptions in the brain are not view-based.

Contents

Chapter 1

Introduction

For most of us, vision is one of the primary senses we use to perceive the world. Our vision system is, in fact, so well-adapted to the task that we often do not realize the complex problems involved in the process of understanding the world around us. Consider the following example: your favorite technical gadget breaks. You suspect a simple malfunction such as a loose connector and feel confident that you can repair it, even though you are not familiar with the insides of this particular type of device. You therefore decide to carefully take the gadget apart, and while doing so make sure to remember the arrangement of the parts you remove. Already, this is a challenging task for your vision system because you may have never seen these exact parts before. Of course, you may categorically recognize, for example, screws, but the exact types of screws may be unfamiliar to you, and differentiating between them may be important when you reassemble the device. Moreover, to properly handle the screws, you must have a good estimate of their location and orientation in space, as well as their size and boundaries. To help you in your repair task, you may carefully lay out each component on your work surface, using space to help you remember how to reassemble the device. By moving components around, you will see them in positions and orientations in which you have never seen them before. But even then, you will likely be able to reidentify them. You will also be able to recognize the components if the lighting conditions change, say, if your repair takes long enough for the sun to start setting, even though this may drastically change the appearance of the objects. This is again true not only for classes of components that you have experienced before, but also for individual components that are entirely unfamiliar to you.

The example illustrates powerful capacities of the human vision system. Being able to reproduce these in artificial cognitive systems is desirable, but requires an understanding of how they come about. Unfortunately, despite

extensive knowledge of the anatomical structure of the vision system in humans and other primates, we have not yet reached such an understanding, and reproducing the brain's capabilities for visual perception in artificial systems thus remains an unsolved problem (Kourtzi and Connor, 2010; DiCarlo et al., 2012; Krüger et al., 2013).

Early attempts to understand biological vision were concerned with questions about the representation of objects in the brain, and how this representation may be matched to visual input. Marr and Nishihara (1978), for example, argue that objects are represented in an object-centered coordinate frame that has to be aligned with the object in the input image to recognize it. The authors further argue that objects are represented as a combination of smaller parts (also referred to as a structural description), a concept later extended by Biederman (1987) into the *recognition by components* framework. In this framework, objects are represented as an arrangement of geometric primitives called *geons*. Recognizing an object in an image therefore means finding the geons it contains and matching their arrangement to one of the stored representations.

Later, evidence arose that the brain represents different view points of objects rather than representing each object by a single structural description (for a review, see Peissig and Tarr, 2007). This evidence lead to the development of the influential HMAX model of the visual cortex (Riesenhuber and Poggio, 1999). Inspired by the findings of Hubel and Wiesel (1962, 1968), HMAX postulates alternating layers of simple and complex cells. The first layer consists of simple cells which are sensitive to edges of specific orientations in small subregions of the retina. The second layer consists of complex cells which pool over small regions of the output of the simple cells, passing only the maximal activation values in the pooled area to the next level of the hierarchy. Over the next layers, the receptive fields of the cells increase in size, and the stimuli that induce maximal responses in the cells become increasingly complex. In the highest level of the hierarchy, *view-tuned cells* respond maximally to specific object views over the whole retina (though later models based on HMAX, for example Serre et al., 2007, do not use view-tuned units).

Though the HMAX model initially dealt mostly with the recognition of computer generated images showing bent wires, it has since been extended to perform classification tasks by learning more complex features (Serre et al., 2004), and has achieved high performance in real-world classification scenarios (Serre et al., 2004, 2007). Backed by experimental evidence, it is viewed as the 'standard model' of the early (feedforward) stages of visual processing (Riesenhuber and Poggio, 2003; Serre et al., 2004) and provides support for the predominantly feedforward picture of visual processing. Further evi-

dence for the feedforward picture comes from studies showing that humans can detect stimuli of certain classes with exposure durations as short as 20 milliseconds, and that visual processing in these cases takes less than 150 milliseconds (Thorpe et al., 1996). Arguably, such a time constraint is too short to allow for top-down influences based on slower recurrent processes. Motivated, in large part, by the evidence reviewed here, many artificial vision systems have adopted the feedforward picture (Fukushima, 1980; LeCun and Bengio, 1995; LeCun et al., 1998; Riesenhuber and Poggio, 1999; Serre et al., 2007; Wiskott and Sejnowski, 2002).

Recently, feedforward image processing is dominated by deep convolutional networks (LeCun et al., 1998; Hinton et al., 2006; Bengio et al., 2007). The structure of these networks shares similarities with the HMAX model (Schmidhuber, 2014), and they have achieved performance similar to and even exceeding that of humans (Cireşan et al., 2012). However, their biological plausibility is questionable (Schmidhuber, 2014).

Though the feedforward picture appears to be a good model for the brain's categorization process, it does not address all capabilities of the human vision system. For example, fast recognition as described in Thorpe et al. (1996) does not include subordinate level information, nor does it include information on the location or feature values of the perceived object (Mack and Palmeri, 2011). Feedforward models usually also require extensive training because class invariances must be learned from many examples (though exceptions exist, for example Fei-Fei et al., 2003). This contrasts with the ability of humans to learn to recognize individual object instances even in poses in which they were not perceived before, from just a small number of training views (for example, Jolicoeur, 1987; Nazir and O'Regan, 1990).

A model that addresses some of these issues is described by Faubel and Schöner (2009, 2010) and Lomp et al. (submitted 2016). The model is capable of recognizing individual objects and estimating their pose based on a small number of training views. Objects are represented by histograms of color and edge orientations as well as a heuristic shape description. During recognition, the system concurrently determines the best matching learned view for the current input, as well as the transformation between this view and the input. The model is based on dynamic field theory (DFT; Schöner, 2008; Schöner et al., 2015b), a framework for building neural-dynamic architectures that has been used widely to model cognitive processes (for example., Johnson et al., 2009; Zibner et al., 2011b; Schneegans and Schöner, 2012). At the core of DFT are dynamic neural fields which describe the activation of populations of neurons over metric feature spaces. In the object recognition system, these feature spaces are the parameter values of the transformation between learned object views and the current input. Each learned view is indexed

by a label, and the activation of the labels is represented by dynamic neural nodes, a variant of the dynamic field equation.

The model provided a first neural-dynamic implementation of the principles of the map-seeking circuit introduced by Arathorn (2002). As such, its aim was to prove that the iterative model of Arathorn (2002) could be realized by the continuous-time differential equations underlying dynamic field theory, and to prove that such a model could achieve sufficient performance in the context of a robotic vision system.

In my thesis, I develop this model further toward biological plausibility. To do so, I address two aspects that were previously not solved in a neurally plausible way. The first concerns the sequential scanning of multiple objects. Thus far, external control, either by the user or an algorithmic structure, is necessary to transition from recognizing one object to recognizing the next. In Part II, I address this by integrating the object recognition system with a scene representation system (Zibner et al., 2011b; Zibner and Faubel, 2015), which models the biological processes of sequentially attending salient locations in the input image and committing their feature values to working memory using the principles of dynamic field theory. In the combined system, attention from the scene representation model determines the input for the object recognition system, which in turn provides identity information to be stored in working memory. Dynamic field theory provides tools for achieving such an integration. However, this requires that the involved architectures are fully realized within the DFT framework. For the scene representation, this is the case, but the transition from recognizing one object to recognizing the next is solved outside this framework in the object recognition architecture presented in Faubel and Schöner (2009, 2010) and Lomp et al. (submitted 2016). Part of the integration therefore consists of replacing these nonneural aspects by mechanisms realized within the DFT framework.

Second, I address the representation of object views. Thus far, the model mainly uses histograms of feature values for recognition. Though most of the mechanisms for extracting these histograms can be implemented neurally, the representation itself deviates from the principles of dynamic field theory (they are not stable distributions of activation over a feature space). In Part III, I therefore introduce representations that are closer to these principles.

An advantage of this new representation is that it allows the system to estimate scale, which was not estimated by the histogram based system. As I describe in Part III, the new representation also reveals additional constraints for the neural dynamics of the object recognition process. In particular, the new representation is less discriminative than the histograms of the original architecture during the initial stages of recognition. This results in ambi-

guities which may lead the dynamics of the system to converge to wrong estimates, effectively resulting in worse recognition performance. I address this by introducing competition in the neural dynamics that gradually increases over the time, which brings the process closer to the competition function central to the map-seeking circuit (Arathorn, 2002). The new object representation also allows to distinguish between foreground and background locations in the input image. I describe how such a distinction may arise from the object recognition system, qualitatively demonstrate it in the presence of occlusion, and investigate its usefulness for masking the input.

Previous evaluations of the object recognition system mainly focused on recognition and pose matching performance in a robotic context. In my thesis, I adopt this setting, but perform further evaluations to characterize how the system's behavior depends on the pose of the object in the image. I relate these results to the literature, in particular on view-based versus structural descriptions and on mental rotation mentioned above in order to draw conclusions about the suitability of the model to explain part of the brain's vision system.

Part I

Dynamic field theory and cognitive vision

Chapter 2

Architectures in dynamic field theory

The architectures I present build on dynamic field theory (DFT; Schöner, 2008; Schöner et al., 2015b), a modeling language for cognitive architectures that is based on the principles of neural dynamics (Grossberg, 1978). In this chapter, I formally describe the relevant principles of DFT. I start with its core elements, dynamic neural fields and nodes, then describe how multiple instances of these elements can be coupled to form larger architectures, and how the stable states in these architectures may be organized to fulfill an intended function. I then describe the numerical implementation of DFT that I use and, finally, introduce some notation aimed at simplifying the equations throughout the rest of this thesis.

Please note that some of the text in the present chapter is based on Lomp et al. (accepted 2016).

2.1 Dynamic neural fields and nodes

At the core of the DFT framework are *dynamic neural fields* (which I also refer to as *fields*), which describe how the distribution of activation, $u(\mathbf{x}, t)$, of a population of neurons evolves in time, $t \in \mathbb{R}^+$. Activation is defined over a feature space indicated by $\mathbf{x} \in \mathbb{R}^n$, which defines the field's dimensionality, $n \in \mathbb{N}$. The feature space can, for instance, be a one-dimensional representation of color ($\mathbf{x} = c$ where c is a color) or a two-dimensional location on the retina ($\mathbf{x} = (x, y)$). The activation of a field evolves according to the

dynamics

$$\begin{aligned}\tau \dot{u}(\mathbf{x},t) = &- u(\mathbf{x},t) + h + w_\xi \; \xi(\mathbf{x},t) \\ &+ s(\mathbf{x},t) \\ &+ \left[k^{\mathrm{u}} * (g \circ u)\right](\mathbf{x},t), \end{aligned} \tag{2.1}$$

where τ is a timescale that determines the speed with which activation changes, $s(\mathbf{x},t) \geq 0$ is input to the field which may come from sensors or other fields (see below), $h \leq 0$ indicates the resting level to which activation relaxes in the absence of input ($s(\mathbf{x},t) = 0 \; \forall \mathbf{x}$) and $\xi(\mathbf{x},t)$ is a Gaussian white noise term with unit variance which is scaled by the factor $w_\xi \geq 0$. The last term in the equation brings about interaction in the field by convolving the field's output, $g\left(u(\mathbf{x},t)\right)$, and the interaction kernel, k^{u},

$$\left[k^{\mathrm{u}} * (g \circ u)\right](\mathbf{x},t) = \int k^{\mathrm{u}}(\mathbf{x}-\mathbf{x}') g\left(u(\mathbf{x}',t)\right) \; d\mathbf{x}' \tag{2.2}$$

The interaction kernel may be realized as a combination of Gaussians,

$$k^{\mathrm{u}}(\mathbf{x}-\mathbf{x}') = \sum_i a_i \; G_{\sigma_i}(\mathbf{x}-\mathbf{x}') - \gamma, \tag{2.3}$$

where $\gamma \leq 0$ defines global inhibition that acts on all field sites, and G_{σ_i} is a Gaussian function centered around zero that has width σ_i for dimension i,

$$G_{\sigma_i}(\mathbf{x}-\mathbf{x}') = \exp\left(-\frac{(\mathbf{x}-\mathbf{x}')^2}{2\sigma_i^2}\right). \tag{2.4}$$

I use a "Mexican hat" kernel which comprises two Gaussians: one expresses local excitation between neighboring sites with strength $a_1 \geq 0$ in a region of width σ_1, and the other expresses local inhibition with strength $a_2 \leq 0$ around this region with width $\sigma_2 > \sigma_1$.

The output of the field lies between zero and one and determines which sites of the field contribute to interaction. I use two variants for the output function, $g(u)$. One is a computationally efficient approximation of the logistic function,

$$g(u) = \sigma_{\mathrm{abs}}(u) = \frac{1}{2}\left(1 + \frac{\beta u}{1+\beta|u|}\right), \tag{2.5}$$

where the parameter $\beta > 0$ controls the steepness of the function around zero. I also use the Heaviside function,

$$H(u) = \begin{cases} 1: & u > 0 \\ 0: & u \leq 0 \end{cases}, \tag{2.6}$$

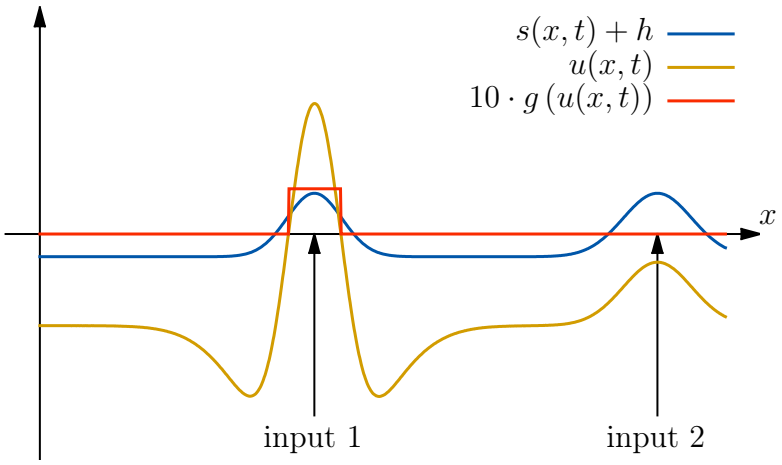


Figure 2.1: An example of a neural field defined over a single dimension, x, and its input (plotted relative to the resting level as $h + s(x,t)$), activation ($u(x,t)$) and output ($g\,(u(x,t))$, here multiplied by a factor for improved visibility). Two equally strong localized inputs are present. The field has formed a suprathreshold (activation above zero) peak over the left input, marked *input 1*. Via global inhibition, this peak suppresses the right input, marked *input 2*, and the field is therefore said to be selective.

---

which may be seen as a computationally efficient approximation of the limit case $\beta \to \infty$ in Equation 2.5.

The field dynamics were introduced by Amari (1977), who showed that localized regions of suprathreshold activation called *peaks* (see Figure 2.1) are a stable solution for an appropriate choice of parameters. Peaks arise in response to localized inputs, and their location encodes feature values. They are thus the elementary unit of representation in dynamic field theory and this kind of representation is referred to as space code.

Dynamic neural nodes (which I also refer to as nodes) are, in a sense, zero-dimensional fields, that is, their feature space is reduced to a single point, and their activation is therefore reduced to a single dynamical variable. Analogous to Equation 2.1, the dynamics of this variable is governed by

$$\begin{aligned} \tau \dot{u}(t) = &- u(t) + h + w_\xi\, \xi(t) \\ &+ s(t) \\ &+ w^{\mathrm{u}}\, g\,(u(t))\,, \end{aligned} \tag{2.7}$$

where, again, $h \leq 0$, $s(t) \geq 0$, $w_\xi \geq 0$, and $\xi(t)$ is a Gaussian white noise term with unit variance. The lack of a feature space implies that there is no explicit metric to relate such nodes to each other. The only interaction in the

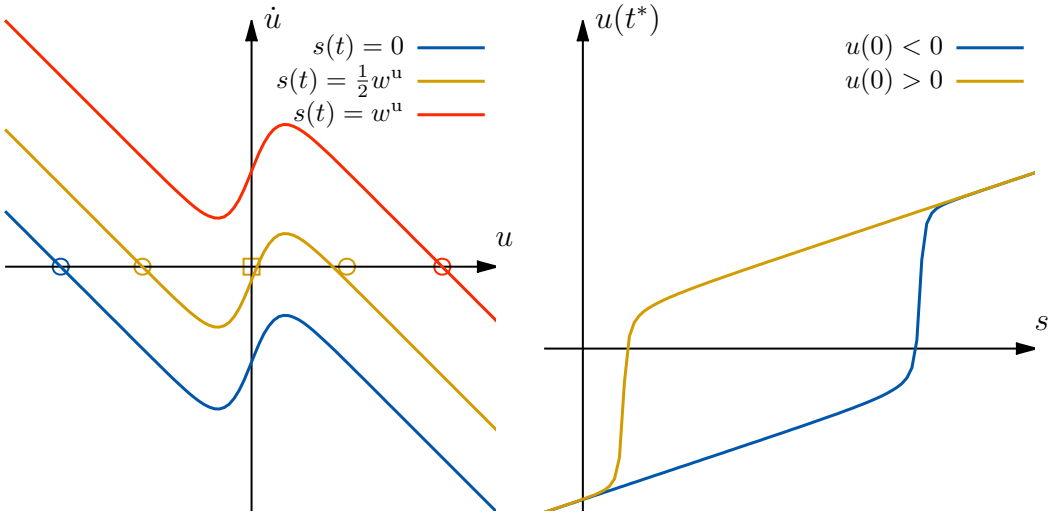


Figure 2.2: (a) shows the phase plot of a node for varying input strengths. Attractors are marked with circles and the repellor is marked with a square. (b) probes the attractors of the node for different input strengths by showing the activation of the system after a time t^* that allows for sufficient convergence, starting from different initial conditions.

---

equation is therefore self-excitation of strength $k^{\mathrm{u}} \geq 0$. Note, however, that multiple nodes can interact with each other through the input, $s(t)$. Such couplings are the topic of the next subsection.

2.1.1 Instabilities in fields and nodes

The functional states of fields and nodes are attractors, solutions of the dynamics for which the rate of change, $\dot{u}$, is zero, and to which nearby solutions converge.

First, consider the dynamics of a node (Equation 2.7). Attractors can be calculated analytically, but they may also be seen directly in the phase plot of the dynamics (Figure 2.2(a)) as the locations at which the graph intersects the u-axis. As Figure 2.2(a) illustrates, different input levels lead to different configurations of attractors. For zero input ($s(t) = 0$), only a single attractor exists at an activation level below threshold. In this case, all solutions converge to it: if the state of the node, u, is below the attractor, the rate of change is positive, and the state therefore changes toward the attractor over time; conversely, if the state is above the attractor, the rate of change is negative, and the state again moves toward it. Thus, no matter

in which state the system starts, it will always converges to a subthreshold state (a state in which the node is said to be *inactive* or *off*). At moderate levels of input ($s(t) = \frac{1}{2}w^{\text{u}}$), there are three fixed points. Two are attractors, one of which is an inactive state (< 0) and one of which is an active state (> 0). A repellor, which also has a rate of change of zero but from which solutions diverge, delineates the basins of attraction between the two states. States below this unstable point converge toward the inactive (*off*) state, whereas states above it converge to the active (*on*) state. For high levels of input ($s(t) > w^{\text{u}}$, the system again shows only a single attractor, this time an *on* state.

How does the system transition between these different configurations of stable states? This may be investigated by continuously varying the parameter relevant to these changes over time and plotting the attractors for each configuration. The relevant parameter here is the strength of input, and the corresponding plot is shown in Figure 2.2(b), which shows that a single attractor "splits" into multiple attractors as the input becomes larger. At high levels of input, multiple attractors merge together. The exact points at which the number or stability of fixed points changes are called *bifurcations* or *instabilities*, and the parameter being varied (in this case the input strength) is called the bifurcation parameter.

Instabilities play an important role in dynamic field theory. They mark discrete events in a system that evolves in continuous time. In the example above, we can observe two such instabilities. The first is the *detection instability*. It occurs at the level of input at which no *off*-state is stable, and the only remaining fixed point corresponds to an *on*-state. The *reverse detection instability* describes the opposite case; it occurs for levels of input at which no *on*-state is stable and the only remaining fixed point corresponds to an *off*-state.

Between these instabilities, the system is bistable, that is, both the *on*- and *off*-state coexist as stable solutions. This is an important property of the system, which can be illustrated by a system that starts with zero input and in an *off*-state. If input is slowly increased, the system remains below threshold until the detection instability occurs, at which point the input is said to be detected. When input is lowered again, the system remains in the active state, even when the level of input goes below the level that induced the initial detection. This is considered an elementary cognitive act: the 'decision' to detect the input is stabilized against fluctuations by the *hysteretic* properties of the system. The system becomes inactive only when the level of input drops significantly, to the point at which the reverse detection instability occurs.

How strongly the system resists fluctuations in the input level depends,

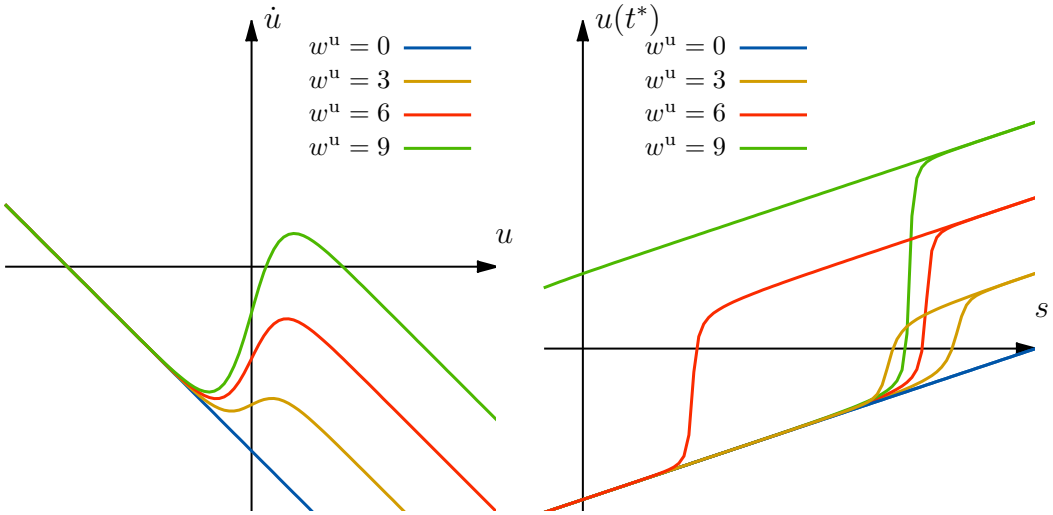


Figure 2.3: (a) shows the phase plot of a node for $s(t) = 0$ and different levels of self-excitation. At high levels of self-excitation, the node may be bistable even without input. (b) demonstrates this by plotting the approximated fixed points of the node for different levels of self-excitation (Figure 2.2 explains how these are obtained).

---

in part, on the node's self-excitation, w^{u}. This can be visualized by a looking at the stable states that arise when self-excitation is varied. An exemplary plot of this is shown in Figure 2.3. As the figure demonstrates, increasing self-excitation lowers the input level for which the reverse detection instability occurs. Above a critical level of self-excitation, the reverse-detection no longer occurs for valid (that is, nonnegative) input levels. This means that if the node becomes activate, it will not deactivate even if the input is set to zero. The node is therefore said to have *memory* of the detection that initially activated it.

To analyze the stable states of dynamic neural fields, a more complex mathematical analysis is necessary. Such an analysis may be found in Amari (1977). Of interest for my thesis are *peak* solutions which are localized connected regions of suprathreshold activation form in response to localized inputs (see Figure 2.1). Analogs for the detection and reverse detection instabilities occur for peaks. The detection instability can be induced by gradually increasing the strength of localized input. As long as the input strength is small, the field relaxes to a subthreshold attractor which has the shape of the input, offset by the resting level, analogous to the sub-threshold solution of the node dynamics. Once the input is strong enough, the detection instability

occurs. Activation in parts of the field becomes suprathreshold, and interaction takes effect. Self- and local excitation stabilize the peak analogous to the self-stabilization of the node. When lowering the input strength, the peak again persists even for input strengths below the one originally inducing detection. Finally, the peak decays when the reverse detection instability occurs.

Neural fields may also exhibit (working) memory. With sufficient local excitation, a peak that is formed is stabilized sufficiently so that it remains active even if the inducing input disappears.

Fields exhibit another instability that is not present in (uncoupled) nodes. When the interaction contains strong global inhibition, peaks strongly inhibit other field sites. When the input contains multiple locations that are sufficiently strong to push the field above threshold (an example is illustrated in Figure 2.1), both locations may initially induce suprathreshold activation in the field. Because this means that they both contribute to interaction, they also inhibit each other. One location them may reach higher activation due to random fluctuations such as noise and thus inhibits the other location more strongly. Over time, this advantage amplifies, and activation at the losing site falls below threshold. Effectively, the field has performed a *selection* of one of multiple candidates (Figure 2.1 illustrates the result of such a selection).

2.2 Coupling neural fields and nodes to form architectures

Individual fields and nodes already have cognitive functions such as detection, selection and memory. Models for more complex cognitive functions, however, require multiple fields and nodes that are coupled together into an architecture (Zibner et al., 2011b; Zibner and Faubel, 2015). In this section, I describe the principles for these couplings using two fields, a source field indexed with A, and a target field indexed with B. The source field has dimensionality a, and the target field has dimensionality b. Field A is coupled to field B by providing input $s^{\mathrm{B}}(\mathbf{x}, t) = s^{\mathrm{B,A}}(\mathbf{x}, t)$ (note that s^{B} may be a sum of multiple coupling terms and other inputs; for simplicity, I do not include this in the notation here).

In the *one-to-one* coupling, both fields have the same dimensionality ($a = b$). The input to the target field is the output of the source field, $g\left(u^{\mathrm{A}}(\mathbf{x}, t)\right)$,

convolved by a coupling kernel, $k^{\mathrm{B,A}}(\Delta\mathbf{x})$,

$$\begin{aligned} s^{\mathrm{B,A}}\left(\mathbf{x},t\right) &= \left[\left(g\circ u^{\mathrm{A}}\right) * k^{\mathrm{B,A}}\right]\left(\mathbf{x},t\right) \\ &= \int k^{\mathrm{B,A}}(\mathbf{x}-\mathbf{x}')g\left(u^{\mathrm{A}}(\mathbf{x}',t)\right)\ d\mathbf{x}'. \end{aligned} \tag{2.8}$$

This assumes that both fields are defined over the same dimensions, that the dimensions are aligned, and thus that the fields are defined over the same vector, $\mathbf{x}$. The coupling kernel, $k^{\mathrm{B,A}}$, may, for example, be a zero-mean Gaussian (see Equation 2.4). Note that the kernel is assumed to be independent of time, and the convolution is assumed to disregard the time argument in any such couplings.

When the target field represents more metric dimensions than the source field ($b > a$), the coupling is an *expansion*. In this case, the vector $\mathbf{x}^{\mathrm{B}}$ that describes the dimensions of the target field contains all dimensions present in the vector $\mathbf{x}^{\mathrm{A}}$ that describes the dimensions of the source field, but has additional entries. The input to the target field is

$$\begin{aligned} s^{\mathrm{B,A}}\left(\mathbf{x}^{\mathrm{B}},t\right) &= \left[\left(g\circ u^{\mathrm{A}}\right) * k^{\mathrm{B,A}}\right]\left(\mathbf{x}^{\mathrm{A}},t\right) \\ &= \int k^{\mathrm{B,A}}(\mathbf{x}^{\mathrm{A}}-\mathbf{x}')g\left(u^{\mathrm{A}}(\mathbf{x}',t)\right)\ d\mathbf{x}'. \end{aligned} \tag{2.9}$$

Because the right hand side does not depend on the extra dimensions in $\mathbf{x}^{\mathrm{B}}$, the input is constant along these dimensions. Zibner et al. (2011b) describe this as *ridge inputs* for two-dimensional target fields and one-dimensional source fields, as tube inputs for three-dimensional target fields and two dimensional source fields, and as slice inputs for three-dimensional target fields and one-dimensional source fields.

When the target field represents fewer metric dimensions than the source field ($b < a$), the coupling is a *contraction*. In this case, some of the dimensions represented by $\mathbf{x}^{\mathrm{A}}$ are not represented by $\mathbf{x}^{\mathrm{B}}$. For notational convenience, assume that the first b dimensions are shared between both vectors, and that the extra dimensions are in the last slots, $x^{\mathrm{A}}_{b+1},\ldots,x^{\mathrm{A}}_{a}$. These dimensions are contracted by integration,

$$s^{\mathrm{B,A}}\left(\mathbf{x}^{\mathrm{B}},t\right) = \int\cdots\int g\left(u^{\mathrm{A}}(\mathbf{x}^{\mathrm{A}},t)\right)\ dx^{\mathrm{A}}_{b+1}\ \ldots\ dx^{\mathrm{A}}_{a}. \tag{2.10}$$

Note that the result may be further convolved with a coupling kernel; I leave this out in the equation to prevent notational clutter.

A source node indexed with A and with activation $u^{\mathrm{A}}(t)$ may be coupled to a target node indexed by B by providing coupling input $s^{\mathrm{B}}(t) = s^{\mathrm{B,A}}(t)$

(note that again, s^{B} may be a sum of multiple coupling terms and other inputs). This input is given by

$$s^{\mathrm{B,A}}(t) = k^{\mathrm{B,A}}\ g\left(u^{\mathrm{A}}(t)\right), \tag{2.11}$$

where $k^{\mathrm{B,A}} \in \mathbb{R}$ is the strength of the coupling.

When the source node A is coupled to a target field B, it may provide a boost to the entire field:

$$s^{\mathrm{B,A}}(\mathbf{x}, t) \;=\; k^{\mathrm{B,A}}\ g\left(u^{\mathrm{A}}(t)\right), \tag{2.12}$$

where $k^{\mathrm{B,A}}$ is the strength of the coupling. Such a coupling may be used to induce a detection instability. The coupling strength may also be given by a function that depends on space, $k^{\mathrm{B,A}}(\mathbf{x})$, so that the source node induces a localized input in the target field. For example, a 'red' node may have a Gaussian-shaped connection function that is centered around the value for red.

When a source field A is coupled to a target node B, the coupling input is analogous to a contraction of all field dimensions,

$$s^{\mathrm{B,A}}(t) \;=\; k^{\mathrm{B,A}} \int g\left(u^{\mathrm{A}}(\mathbf{x}, t)\right)\ d\mathbf{x}. \tag{2.13}$$

When the coupling strength, $k^{\mathrm{B,A}}$, and the parameters of the node are chosen appropriately, a sufficiently strong (and broad) peak in the field A induces a detection instability in the node. The node is thus a *peak detector*. Throughout my thesis, I refer to the activation of such peak detectors by $u^{\mathrm{pd}}_{\mathrm{idx}}(t)$, where idx indexes the field for which the node detects peaks, and the index, pd, indicates that the node is a peak detector.

2.3 Behavioral organization

Complex cognitive models may place constraints on the temporal organization of the stable states of activation of the model's elements. As an example, think of a robot trying to grasp an object on a table. First, the robot must detect the object, for example by forming a peak in a field defined over space. This peak may then drive the motor system to move the robot's hand towards the target. Simultaneously, a one-dimensional field defined over the state of the robot's hand may control the opening and closing of the hand. To grasp an object, it is necessary to open the hand before attempting to grasp. Therefore, a peak at a specific position in the hand-state field may need to be

present before moving the hand towards the target. Such constraints must be realized by controlling instabilities and fields.

Richter et al. (2012) propose a system for *behavioral organization* that realizes such constraints by controlling the instabilities of fields and nodes through *elementary behaviors*.[1] Each behavior consists of two dynamic neural nodes. The *intention node* represents the intention to execute a behavior. The *condition of satisfaction node* signals that the behavior is complete. The intention node may be associated with an intention field which contains a metric representation of the intention (for example, the intention to create a peak representing an open hand), and the condition of satisfaction node may be associated with a condition of satisfaction field that represents completion of the behavior by forming a peak.

Next, I formally define such behaviors and the dynamics of their nodes and fields. I then describe how two kinds of constraints between behaviors may be realized. A *suppression* prevents two or more behaviors from becoming active at the same time. A *precondition* ensures that a behavior only activates if another has already been completed. For the formalizations, I adopt the framework proposed by Richter et al. (2012).

2.3.1 Dynamics of a single behavior

Each behavior is activated by a *task* input, $s^{\text{task}}(t) \in \{0, 1\}$. The value of this input is determined externally, for example, by a user or a program controlling the architecture. As defined above, a behavior comprises two nodes: the *intention node*, whose activation I denote by $u^{\text{int}}(t)$, and the *condition of satisfaction node*, whose activation I denote by $u^{\text{cos}}(t)$. It may also comprise two fields: the *intention field*, whose activation I denote by $u^{\text{intf}}(\mathbf{x}, t)$, and the *condition of satisfaction field*, whose activation I denote by $u^{\text{cosf}}(\mathbf{x}, t)$, where $\mathbf{x}$ is a feature space as defined for the field equation (see Equation 2.1).

When the *intention node* is active, this signals that a behavior should be enacted (what this means concretely depends on the architecture; it may, for example, mean that a robot should start moving towards a target). The

[1] Note that in the following, I use the term *behavior* instead of *elementary behavior* because in the context of the architectures I present, behaviors may drive other behaviors and thus are no longer elementary.

node's dynamics is analogous to the dynamic node equation (Equation 2.7):

$$\begin{aligned}\tau \dot{u}^{\text{int}}(t) = &- u^{\text{int}}(t) + h + w_\xi \, \xi(t) \\ &- w^{\text{int,cos}} \, g\left(u^{\text{cos}}(t)\right) + s^{\text{int}}(t) + w^{\text{int,task}} s^{\text{task}}(t) \\ &+ w^{\text{int}} \, g\left(u^{\text{int}}(t)\right). \end{aligned} \tag{2.14}$$

Here, $s^{\text{int}}(t)$ stands for a collection of inputs from the architecture in which the behavior is embedded. This input may be inhibitory, preventing activation of the node from becoming suprathreshold, for example, to realize constraints between behaviors (see Section 2.3.2).

The intention node may excite the *intention field*, which is governed by the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{intf}}(\mathbf{x}, t) = &- u^{\text{intf}}(\mathbf{x}, t) + h + w_\xi \, \xi(\mathbf{x}, t) \\ &+ k^{\text{intf,int}}(\mathbf{x}) \, g\left(u^{\text{int}}(t)\right) + s^{\text{intf}}(\mathbf{x}, t) \\ &+ \left[k^{\text{intf}} * \left(g \circ u^{\text{intf}}\right)\right](\mathbf{x}, t), \end{aligned} \tag{2.15}$$

where $s^{\text{intf}}(\mathbf{x}, t)$ represents input that comes from other fields in the architecture and $k^{\text{intf,int}}(\mathbf{x})$ is a pattern that encodes the intended outcome of the behavior. For an example, think of a 'look for red' behavior. The intention field may be defined over color, and $k^{\text{intf,int}}(\cdot)$ may be a Gaussian centered on the value for red.

The *condition of satisfaction node* has the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{cos}}(t) = &- u^{\text{cos}}(t) + h + w_\xi \, \xi(t) \\ &+ w^{\text{int,int}} \, g\left(u^{\text{cos}}(t)\right) \\ &+ s^{\text{cos}}(t) + w^{\text{cos,task}} s^{\text{task}}(t) \\ &+ w^{\text{cos}} \, g\left(u^{\text{cos}}(t)\right). \end{aligned} \tag{2.16}$$

Activation is driven by the input, $s^{\text{cos}}(t)$. With sufficient input, the activation reaches threshold and the behavior is considered complete. The exact equation for the input depends on the context of the behavior; it may, for example, be a sum of outputs from condition of satisfaction nodes of other behaviors, so that the behavior completes only when all connected behaviors are complete as well. If the behavior has a condition of satisfaction field, this input may be the contraction of that field's activation (denoted by u^{cosf}),

$$s^{\text{cos}}(t) = w^{\text{cos,cosf}} \int g\left(u^{\text{cosf}}(\mathbf{x}, t)\right) \, d\mathbf{x}, \tag{2.17}$$

so that the condition of satisfaction node is a peak detector (see Section 2.2).

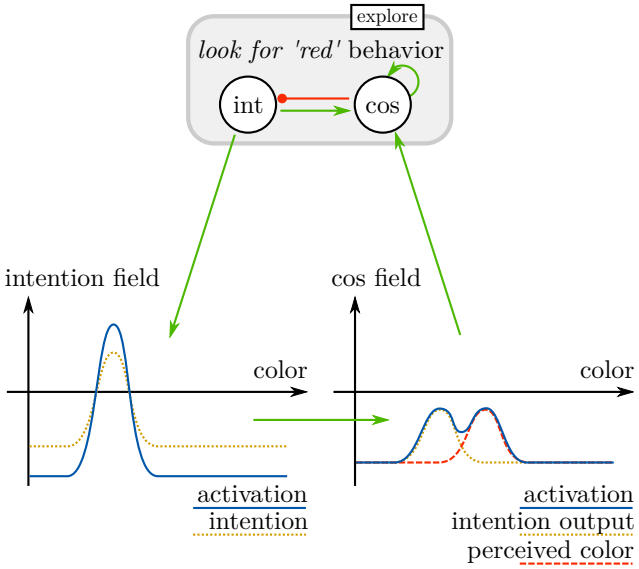


Figure 2.4: Graphical representation of a behavior. The gray box groups the intention (int) and condition of satisfaction (cos) node in the exemplary behavior *look for 'red'*. This behavior is part of a task *explore*, as indicated by the small box at the top-right side of the behavior. Lines ending in an arrowhead indicate an excitatory coupling. Lines ending in a circle indicate an inhibitory coupling.

One way in which the *condition of satisfaction field* may detect completion is by overlaying the current state in the world (given by an input, $s^{\text{cosf}}(\mathbf{x}, t)$) with the intended state (the output of the intention field). Think again of the example behavior 'look for red'; the activation of the intention field represents the color red, and feeds into the condition of satisfaction field. However, the coupling is not strong enough, and on its own does not induce a peak. A peak is induced only when this input overlaps sufficiently with the perceived color. The dynamics of the condition of satisfaction field,

$$\begin{aligned}\tau \dot{u}^{\text{cosf}}(\mathbf{x}, t) = &- u^{\text{cosf}}(\mathbf{x}, t) + h + w_\xi \, \xi(\mathbf{x}, t) \\ &+ w_{\text{cosf}} \left[G_{\sigma_{\text{cosf}}} * \left(g \circ u^{\text{intf}} \right) \right] (\mathbf{x}, t) \\ &+ w_{\text{cosf}} \left[G_{\sigma_{\text{cosf}}} * s^{\text{cosf}} \right] (\mathbf{x}, t) \\ &+ \left[k^{\text{cosf}} * \left(g \circ u^{\text{cosf}} \right) \right] (\mathbf{x}, t) \,, \end{aligned} \tag{2.18}$$

realize this with an appropriately chosen coupling weight, w_{cosf}.

To make the overall structure of a behavior concrete, Figure 2.4 illustrates the exemplary behavior 'look for red'. When the task is activated, the

intention node receives positive input and becomes active. This projects a pattern into the intention field, the Gaussian centered on the color value for red described above. In turn, this induces a peak in the intention field, which projects into the condition of satisfaction field. The intention of the behavior is also assumed to drive the agent controlled by the architecture to explore its environment in a way not represented here. The color of the object in the center of a camera on the agent provides input to the condition of satisfaction field. When the agent perceives red, this input increases activation in the condition of satisfaction field around the value for the perceived color. If this value is close enough to the intended value, it overlaps with the input from the intention, and a peak forms. This peak, in turn, drives the condition of satisfaction node above threshold. The inhibitory coupling from the condition of satisfaction node subsequently deactivates the intention node. The agent therefore stops looking for the color red and may perform further actions such as memorizing that red was found at the current location.

Dedicated intention or condition of satisfaction fields may not necessarily be part of a behavior. Instead, the condition of satisfaction, for example, may be driven by peaks arising in fields that fulfill other roles in the architecture.

It may be necessary to remember that an action was completed even if the condition of satisfaction is no longer perceived. In this case, the condition of satisfaction node may be made self-sustained by strong self-excitation. I indicate this by a loop on the condition of satisfaction node (see Figure 2.4 for an example).

Multiple behaviors may be combined to organize the behavior of an architecture and must be distinguished in the architecture's formalization. Analogous to the notation for fields, I name behaviors, and add an index to the different variables of the behavior. For example, the activation of the intention node of the behavior ***l**ook **f**or **r**ed* may be indexed with 'lfr', and is thus denoted as $u^{\text{int}}_{\text{lfr}}(t)$.

2.3.2 Constraints between behaviors

When an architecture contains multiple behaviors, there may be constraints between these behaviors. For example, two behaviors may drive a common resource such as a robotic arm, implying that the behaviors cannot be active at the same time. In the behavioral organization framework, this is expressed

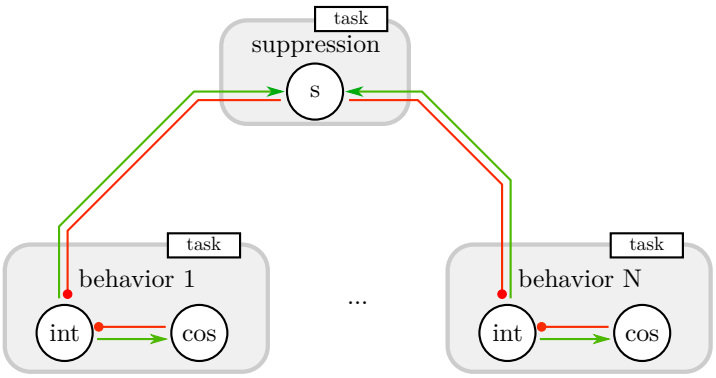


Figure 2.5: Graphical representation of a suppression between competing behaviors.

---

by a *suppression* node which has the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{sup}}(t) = &- u^{\text{sup}}(t) + h + w_\xi \, \xi(t) \\ &+ s^{\text{task}}(t) + w^{\text{sup,int}} \sum_i g\left(u_i^{\text{int}}(t)\right) \\ &+ w^{\text{sup}} \, g\left(u^{\text{sup}}(t)\right), \end{aligned} \tag{2.19}$$

where i indexes competing behaviors and all other variables are defined analogous to the dynamic neural node equation (Equation 2.7). The parameters of the suppression node are chosen such that it activates when at least one of the competing behaviors becomes active. The node inhibits the competing behaviors via an additional input to the dynamics of their intention nodes. The strength of this inhibition, w^{sup}, is chosen so that the intention nodes may not become active if the suppression node's activation is above threshold. This implements selection between the suppressed intention nodes analogous to the selection described in Section 2.1.1. Figure 2.5 shows a graphical representation of nodes that suppress each other.

The second type of constraint, the *precondition*, expresses conditions that have to be met before a behavior may become active. Preconditions are expressed by a *precondition node* whose activation is governed by the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{pre}}(t) = &- u^{\text{pre}}(t) + h + w_\xi \, \xi(t) \\ &- s^{\text{pre}}(t) + w^{\text{task}} s^{\text{task}}(t) \\ &+ w^{\text{pre}} \, g\left(u^{\text{pre}}(t)\right). \end{aligned} \tag{2.20}$$

The coupling strength of the task input, $w^{\text{task}} > h$, is chosen so that the node becomes active as soon as the task activates. When it is active, it inhibits

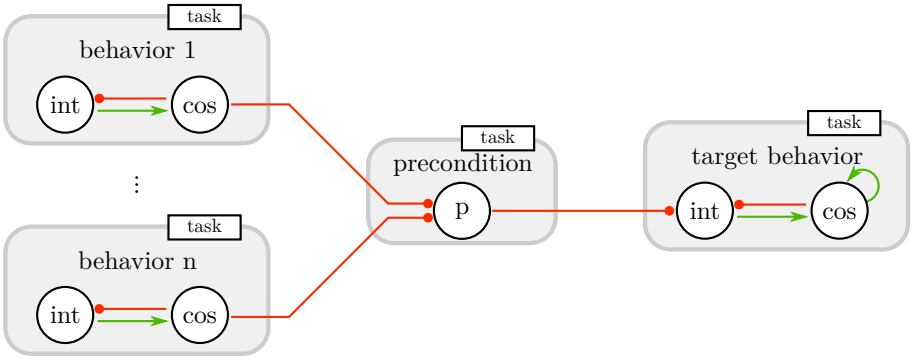


Figure 2.6: Graphical representation of preconditions between behaviors. The precondition node ('p') is activated by the *task* and inhibits the intention node of the target behavior. When the condition of satisfaction nodes of behaviors 1 to n activate, they suppress the precondition node, releasing the inhibition of the target behavior.

---

the intention nodes of the target behaviors of the constraints, preventing them from becoming active. When all of its preconditions are met, the precondition node is inhibited by its input, $s^{\text{pre}}(t)$, and becomes inactive, allowing the intention nodes of the inhibited behaviors to become active. The precondition node's input commonly comprises the activation of condition of satisfaction nodes from other behaviors, u_i^{cos}, for example,

$$s^{\text{pre}}(t) = w^{\text{pre,cos}} \sum_{i=1}^{n} g\left(u_i^{\text{cos}}(t)\right), \tag{2.21}$$

but the exact formulation again depends on the architecture. Different values for the coupling weight, $w^{\text{pre,cos}}$, lead to different types of preconditions. If $w^{\text{pre,cos}} > h$ (where h is the resting level of the precondition node), then a single active condition of satisfaction node in Equation 2.21 suffices to push the activation of the precondition node above threshold. If, on the other hand, $w^{\text{pre,cos}} = \frac{h}{n} + \epsilon$ (where ϵ is a small positive value), the precondition is only deactivated (and thus fulfilled) if all nodes connected to it are active. Figure 2.6 illustrates the connectivity of a precondition node.

2.4 Implementing dynamic field theory

Usually, analytical solutions for the dynamics of a dynamic field theory architecture are not available. To implement the architecture, its dynamics must therefore be approximated numerically. For more complex architectures, this

may mean a significant overhead in terms of the effort required for the implementation, especially if performance is a concern. During my thesis work, I co-developed *cedar* (cognition, embodiment, dynamics, and autonomy in robotics; see Lomp et al., 2013, accepted 2016), a framework that reduces this overhead by offering a graphical interface for constructing architectures from supplied implementations of the core elements of dynamic field theory.[2] All architectures I present are implemented and evaluated using this framework. In the present section, I therefore describe some of the principles underlying the *cedar* framework, in particular, the method for numerically approximating the solutions of the neural dynamics, and the synchronization of real and simulated time necessary in robotic contexts. Please note that some of the descriptions in this section follow Lomp et al. (accepted 2016).

2.4.1 Solving dynamics numerically

cedar numerically approximates the solution of dynamics using the forward Euler method despite its low order of convergence when compared to more sophisticated approaches. This choice was made for several reasons. First, the functional states of dynamic field theory architectures are attractors. To an extent, their stability properties transfer to the numerical approximation, reducing the impact of the Euler method's relatively low order of convergence. Second, when dynamic field theory architectures are embodied, for example, in a robot, the frequency at which the numerical approximation of the neural dynamics is updated is constrained by the frequency at which hardware (for example, a camera) provides new data. This may become problematic for higher order methods. For example, the Runge-Kutta approach may require intermediate steps to determine the update of the approximation. Evaluating the dynamics for these intermediate steps would require a different sampling of the sensor readings, for instance by interpolation. Third, the dynamics implemented in *cedar* (and used for the models presented in my thesis) belong to the class of stochastic differential equations (for example, the neural field equation, Equation 2.1, contains a noise term), and higher-order numerical methods thus require repeated sampling; in fact, they require many function evaluations for each time step (Kloeden and Platen, 1999). Because these evaluations may be costly for neural fields (especially of higher dimensionality), this quickly offsets the computational gain from using higher order methods.

Thus, following the Euler approach, a stochastic differential equation

$$\tau \dot{u}(t) = f(u(t)) + w_\xi \, \xi(t) \tag{2.22}$$

[2] *cedar* is available for download at `http://cedar.ini.rub.de`.

that evolves on a timescale, τ, with deterministic dynamics, $f(u(t))$, and Gaussian white noise, $\xi(t)$, with a variance of one and a mean of zero, multiplied by a factor w_ξ, may be approximated by

$$u(t_i) \approx u(t_{i-1}) + \frac{1}{\tau}\left(\Delta t_i f(u(t_{i-1})) + \sqrt{\Delta t_i}\; w_\xi\; \xi_{i-1}\right), \tag{2.23}$$

with the value from a Gaussian pseudo-random number generator, ξ_{i-1}. The solution is approximated at discrete samples of time, t_i (with $i \in \{1, 2, \ldots\}$). These samples are chosen approximately equidistantly (see below), so that $\Delta t_i = t_i - t_{i-1}$. Note that the stochastic term is scaled with the square root of this time step (see Zwillinger, 1989).

The approximation error of the Euler approach scales with the time step, Δt_i. A small time step is therefore desirable, and the general stability properties of the Euler approach suggest that it should be chosen to be several orders of magnitude smaller than the shortest time scale of the architecture. However, too small values may be smaller than the time required for the computation of the dynamics, which is problematic for reasons I discuss below. In practice, attractor solutions support numerical stability, so that a relatively crude sampling of the dynamics is still possible, with time steps exceeding the limit suggested by the stability of the Euler approach when simulating dynamic field theory architectures.

2.4.2 Synchronizing real and simulated time

Dynamic field theory architectures may be connected to sensors in the real world, for instance, to cameras. In such a case, the time samples, t_i, for which the numerical approximation is calculated, must be aligned with physical time in the real world so that both are properly synchronized. This creates further constraints for the choice of the Euler time step (Δt_i in Equation 2.23).

If the computation of the Euler update systematically takes longer than the chosen Euler time step, real and simulated time inevitably become desynchronized and the architecture cannot be realized appropriately. Assuming that the computation time cannot be reduced (for example, by improving the efficiency of the implementation or using faster hardware), the Euler time step has to be increased so that it is larger than the time required for the computation of updates.

Ideally, the computation of the Euler update (Equation 2.23) is fast enough, and computation time is not a concern. Even so, care must be taken that the physical time at which the updates of the states of the dynamical systems are provided does not become systematically desynchronized with

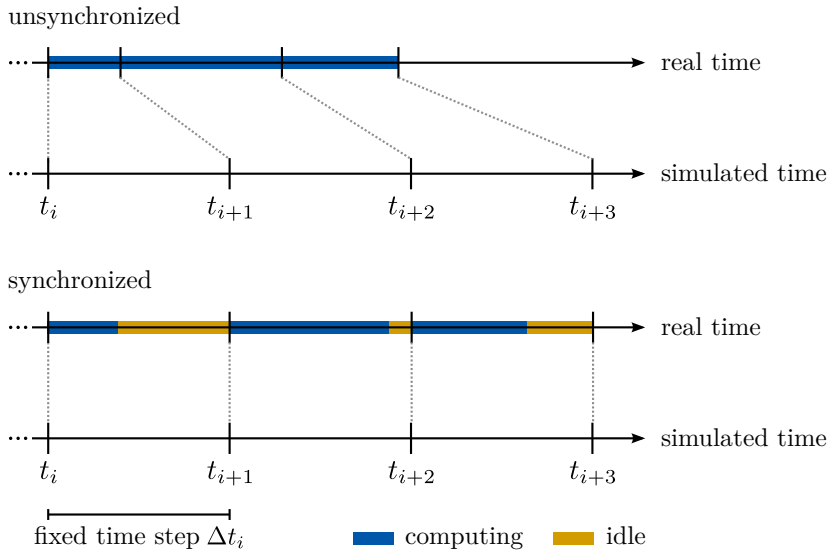


Figure 2.7: An illustration of the synchronization of real and simulated time. Blue bars indicate time at which the Euler step is being actively computed, yellow bars indicate idle times. Computing updates as fast as possible and using the duration of computation as the time step (top panel) may lead to a desynchronization of real and simulated time. Adding idle time so that at least a fixed time has passed addresses this problem (bottom panel).

---

the simulated time. If the evaluation of the numerical solution were started every time the last update was available, less time may have passed in the real world than indicated by the Euler time step (see top panel of Figure 2.7). The solution's time, captured by t_i, would thus increasingly run ahead of physical time, and the properties of the neural dynamics would no longer be correctly realized by the implementation.

The solution offered by *cedar* is to delay the initiation of a new Euler update until enough time has passed in the real world to match the Euler time step (see bottom panel of Figure 2.7). *cedar* also addresses the opposite case, where the computation time takes longer than the Euler time step. In this case, the Euler time step of the next update is extended by the additional time required for the current Euler update. Again, this case should only occur infrequently, for example, due to fluctuations in computation time arising from an environment in which multiple threads of execution interact. *cedar* monitors the occurrence of these events, and their frequency is displayed by a meter that prompts the user to reparameterize the dynamics and lengthen the planned Euler step.

More complex dynamic field theory architectures may include dozens of fields, each with different dimensionalities (for examples, see Zibner et al., 2011a; Richter et al., 2012; Knips et al., 2014). Numerically approximating the dynamics of these architectures in real time may prove challenging. One method for optimizing computational effort is to allow different Euler time steps for different fields. Fields that require longer time to compute their Euler updates (for example, large three-dimensional fields) may then be iterated less frequently than faster fields of lower dimensionality and size.

In *cedar*, architectures may therefore be divided into components that are updated with different Euler time steps. Because the time at which an update is calculated is linked to the Euler time step when the architecture is synchronized with physical time, these components may also be updated with different frequencies, allowing smaller timescales for some of the components. In addition, each such component is its own thread of execution, potentially making use of multiple cores common to modern CPUs.

A downside of this approach is that asynchronies may arise because each thread reads the outputs of other threads at times that may come from time samples that deviate from its own current time. However, synchronization via physical time ensures that all threads of execution always read data from approximately the same time. Asynchronies caused by threading thus do not accumulate and remain small in relation to the time scales in the architecture.

2.4.3 Sampling space

Fields are defined over continuous feature dimensions (see Section 2.1) that must be sampled for the numerical evaluation. In *cedar*, this sampling takes the form of a regular grid. Each node in this grid is its own dynamic equation and implements part of a simple rectangle rule that discretizes the integral in the field equation (Equation 2.1). The dynamics of these grid points are approximated using the Euler approach (Equation 2.23).

The implementation of the rectangle rule for the interaction also involves a discretization of the interaction kernel. *cedar* ensures that the discretized kernel has an uneven size, thus avoiding potential biases. For computational efficiency, convolutions are split up into separable components where possible. In addition, the fast Fourier transform may be used to speed up convolutions when the size or dimensionality of the field or kernel becomes larger.

2.5 Notational conventions

For the precise specification of architectures, I write down the equations for the involved fields and nodes as well as other components. For more complex architectures, notation may become cumbersome because the equations have several parameters which all have to be kept apart. In this section, I introduce some notational conventions that exploit the regularity of the equations in dynamic field theory. These conventions aim to improve the readability of the equations, and I use them throughout the rest of my thesis. An index of the notation may also be found in Appendix A.

When describing field equations, I indicate a specific field by an index on the activation variable. Because architectures may have many fields, this index usually consists of three to four letters indicating the function of the field. For example, $u^{\text{attn}}(\mathbf{x}, t)$ may refer to the activation of an attention field. When writing down the dynamics of this field, all variables need to inherit this index. For example, the field's resting level is h^{attn}. However, I usually suppress this index, instead writing just h. Consequently, variables without any index are meant to be specific to the field, that is, two fields for which the resting level is denoted by h may still have different values for their resting level. If they share the same resting level, I indicate this by an additional index that differs from the field name, for example, h^{shared}.

I indicate the concatenation of functions by the "$\circ$" symbol. This means that $f \circ g(x)$ is equivalent to $f(g(x))$. This is in particular used to describe the convolution with an interaction kernel. Generally, I write the convolution of two functions f and g as

$$[f * g](\mathbf{x}) = \int \cdots \int f(x_1 - x'_1, \ldots, x_n - x'_n)\; g(x'_1, \ldots, x'_n)\; dx'_1 \;\ldots\; dx'_n, \tag{2.24}$$

where $\mathbf{x} = x_1, \ldots, x_n$.

A similar problem arises when fields are coupled. I may express coupling kernels as $k^{\text{tar,src}}$ and connection weights as $w^{\text{tar,src}}$, where 'src' is the index of the field from which the connection originates (the source) and 'tar' is the index of the target field of the connection. However, from the context, it is often clear what the source and target of the connection are, and I thus forgo a special index for the weight term. For example, in the equation

$$\tau \dot{u}^{\text{tar}}(x, t) = \ldots - w\,[G_\sigma * (g \circ u^{\text{src}})\,](x, t)\,, \tag{2.25}$$

it is clear that the activation of the target field receives input from the source field, and the connection weight is thus abbreviated to w, which again follows the convention above in that its value is not shared with other equations.

Note, that these weights are always meant to have a positive sign ($w > 0$), and that the "+" and "−" signs in front of the weights indicate excitatory and inhibitory interactions, respectively. In addition, the convolution disregards the time index (see explanation around Equation 2.2).

A related issue arises when parameters such as connection weights appear multiple times in the same equation. To differentiate these, I index them by numbers, for example, w_1. Again, these parameters are unique to the equation in which they appear. Other equations may reuse w_1, but the values may be different.

I often refer to active and inactive fields and nodes. An active field is one where a peak has formed, that is, a field with a region X_P for which $g\left(u(x_p, t)\right) \approx 1 \; \forall \; x_p \in X_P$ (where u is the field's activation). Inactive fields are those for which such a region does not currently exist. Similarly, a node is active if $g\left(u(t)\right) \approx 1$ (where u is the node's activation) and inactive otherwise. Implicitly, this means that the threshold for activation, determined by the sigmoid function, is always set to zero.

Chapter 3

Object recognition based on localized receptive field histograms

Models for cognitive processes developed in dynamic field theory often simplify visual perception. For example, Johnson et al. (2009) present a model for detecting changes in visual displays in which the input is simplified to Gaussians along a one-dimensional axis representing space. Such simplifications are useful for isolating specific aspects of the cognitive process being modeled, but leave open the question of how the complex processes involved in the brain's visual system may be integrated into such models and dynamic field theory as a whole.

These issues becomes more relevant when dynamic field theory models are embodied in artificial cognitive agents such as robots. In such a context, Faubel and Schöner (2008) developed a model for object recognition based on dynamic field theory. In the model, objects are described by a combination of low-dimensional feature values such as color and aspect ratio. These feature values are represented using space code and are bound together by a shared label dimension in label-feature fields. During a supervised training phase, peaks form in these fields at intersections between the feature values from the input and the label provided by a supervisor. These peaks, in turn, lead to increased activation in a memory trace that stores the information, forming a representation in which a banana, for example, is described as elongated and yellow. When the system recognizes an object in a test image, the label information must be retrieved. Features extracted from the input again provide ridge inputs to the feature dimensions of the label-feature fields. These ridges overlap with the preshape from the memory trace and thus excite specific labels associated with the individual feature values. Over

time, the competitive dynamics of the system settle on a single label, ideally the one with feature values that best match the object in the test image.

A successor to this model is presented by Faubel and Schöner (2009, 2010) and Lomp et al. (submitted 2016). The successor departs from the space coded representation of objects and instead uses localized histograms based on the assumption that they are similar to the high-dimensional feature vectors thought to be represented in visual cortex (Serre et al., 2007). It is capable of learning new objects from a small number of training views and recognizes them while concurrently estimating their pose. Because this successor is the basis for the models I develop in my thesis, I describe it in greater detail over the next sections. Please note that these descriptions follow the ones in Lomp et al. (submitted 2016).

The model represents the estimated pose of an object in the input image using dynamic neural fields. Concretely, position in the image plane is represented by a two-dimensional field, whereas orientation in the image plane is represented in a one-dimensional field. Information on the identity of the matched object is represented by dynamic neural nodes, each of which is associated with one of the object views learned by the system. The dynamics of these fields and nodes form the core of the object recognition approach, and I begin with a sketch of how recognition and pose estimation emerges from these dynamics. A formal description is given in the next sections.

Figure 3.1 illustrates the initial state of the dynamics. Here, the capacity of space code to represent the absence of information is used to indicate that no concrete recognition or matching has taken place yet. All fields are equally (in)active, and so are all nodes representing identity. In the *top-down path* (right column of Figure 3.1), a superposition of learned object views is formed using the approach described by Arathorn (2002). Each view contributes to this memory superposition proportional to the activation of the corresponding identity estimate. In the initial state, the overall identity estimate is unspecific, and all object views therefore contribute equally. Further along the top-down path, the memory superposition is transformed by the current pose estimates. This is achieved again by calculating all possible transformed versions of the memory superposition and then summing them, again weighted by the corresponding pose estimates. In the first stage, the memory superposition is rotated. Because the rotation estimate is unspecific, all orientations again contribute equally, and the rotated superposition contains the memory superposition in all orientations. The rotated superposition is further transformed by the (again unspecific) position estimate to yield a top-down pattern.

In the *bottom-up path*, the input is analogously transformed by the current pose estimate, with the inverse order and direction of the transformations in

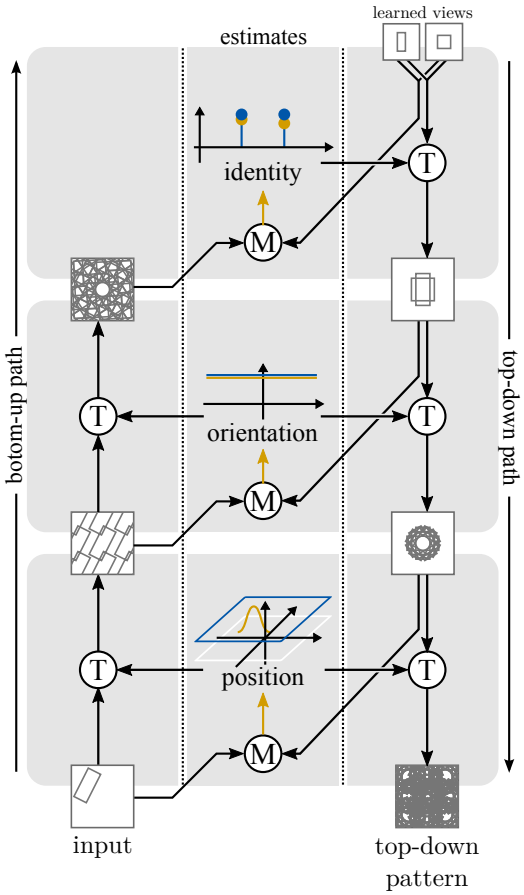


Figure 3.1: An illustration of the object recognition architecture in its initial state. Circles marked with a 'T' indicate pose transformations, while circles marked with an 'M' indicate matching operations. Yellow lines indicate the input to the pose representations, blue lines indicate their current state.

---

the top-down path. Again, since both position estimates are unspecific, all positions and orientations contribute equally, and the bottom-up path yields an unspecific pattern in the initial stage of the recognition process.

Only the matching of position yields specific results in the initial state. In the figure, this is symbolized by Gaussian-shaped input to the position estimate toward which the position representation converges over time. With an increasingly accurate estimate of position, the transformation by the position estimate in the bottom-up path centers the object in the input (see left side of Figure 3.2). This aligns the image with the learned views and allows for more precise matching of the object's orientation. The orientation estimate subsequently starts converging towards the more specific estimate, and

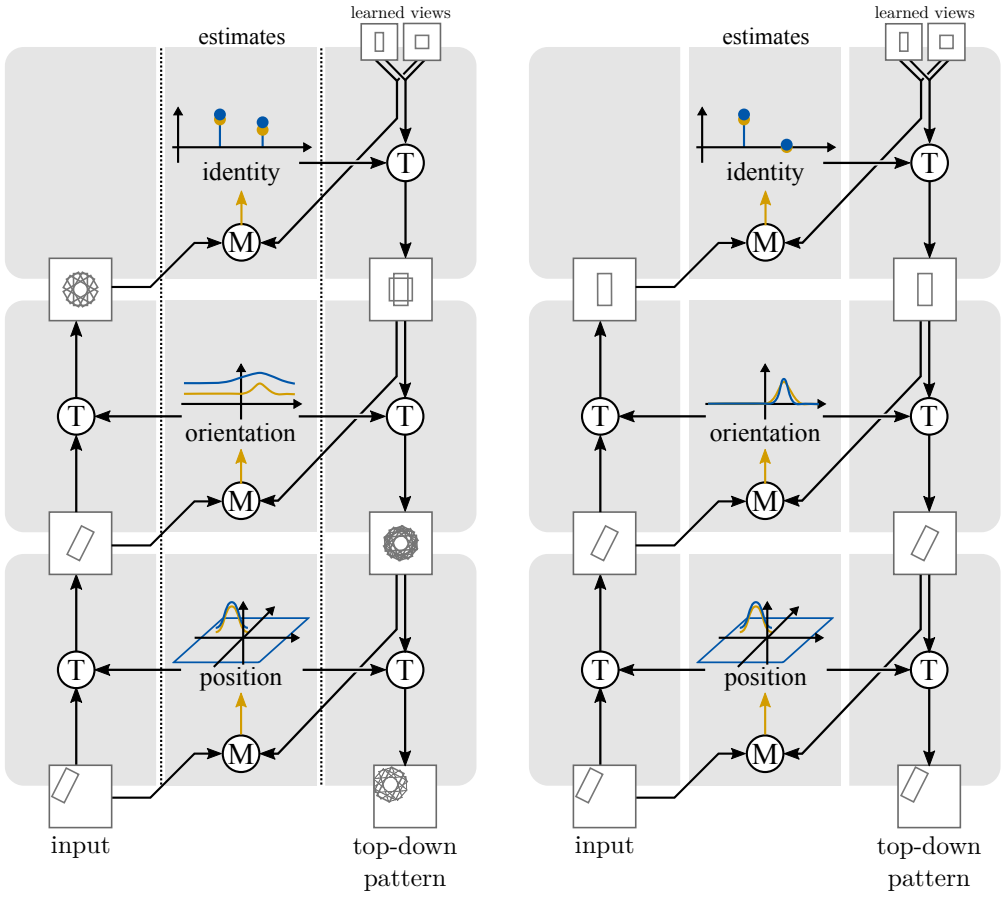


Figure 3.2: Evolution of the state of the object recognition architecture (continued from Figure 3.1).

---

the result of the transformations in the bottom-up path therefore becomes more specific as well (see left side of Figure 3.2). When matched against the learned views, the correct object view thus matches better than the other object view, giving stronger input to the correct label.

This recurrent process continues and, over time, further refines the position, orientation and identity estimates, until it converges to the final recognition decision shown in the right panel of Figure 3.2. In this final state, the pose and label estimates are specific, and the input and top-down patterns are aligned with each other at each stage of processing.

Figure 3.1 and 3.2 only illustrate the shape channel, whereas a total of five feature channels is used in parallel by the full architecture. In the next section, I formalize the transformations in the shape channel. In Section 3.2, I formally describe the neural dynamics of the pose and identity representation.

In Section 3.3, I describe the transformations of the remaining four feature channels. The fusion of evidence from all feature channels into combined pose and identity estimates is described in Section 3.4. Finally, I describe how object views can be learned in Section 3.5.

3.1 The spatial channel

The basis for representing objects in the spatial channel are *spatial patterns*, $I_{\text{sp}} : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}$, extracted from the input image, $I(x, y, t)$, where x and y are image coordinates, and $t \in \mathbb{R}^+$ represents time. Such a spatial pattern could be, for example, the magnitude of the response of an edge filter.

3.1.1 Matching object identity in the bottom-up path

In the bottom-up path, the spatial pattern is first transformed by the current shift estimate, $p_{\text{sh}} \in [0, 1]$ (described in more detail in Equation 3.14),

$$I^{\text{sh}}_{\text{bu}}(x, y, t) = \iint p_{\text{sh}}(x - x', y - y', t)\ I_{\text{sp}}(x', y', t)\ dx'\ dy'. \tag{3.1}$$

This corresponds to the transformation operation in the *position* stage of Figure 3.1 and 3.2. It is analogous to a convolution of the input pattern, I_{sp}, with the current shift estimate, p_{sh}, as kernel. In the sense of Arathorn (2002), this equation can also be seen as the superposition of all possible shifted versions of the input pattern, weighted by the corresponding shift estimate.

To apply rotation, the coordinate system of the transformed pattern is changed to log-polar coordinates. In this coordinate system, where ρ denotes the distance from the origin of the Cartesian coordinate system and ϕ the angle, rotations can be realized as shift operations along the angle dimension. Thus, analogous to Equation 3.1, the rotated version of the shifted pattern is

$$I^{\text{rot}}_{\text{bu}}(\rho, \phi, t) = \int p_{\text{rot}}(\phi - \phi', t)\ I^{\text{sh}}_{\text{bu}}(\rho, \phi', t)\ d\phi', \tag{3.2}$$

where $p_{\text{rot}} \in [0, 1]$ is the current orientation estimate described in Equation 3.14. Figure 3.3 gives an illustration of such an orientation estimate. $I^{\text{rot}}_{\text{bu}}(\rho, \phi, t)$ corresponds to is the transformation in the *orientation* stage in Figure 3.1.

In Cartesian coordinates, the result of Equation 3.2 is the input pattern, rotated and shifted according to current pose estimates.[1] We can now

[1]Note that this transformation to Cartesian coordinates is not strictly necessary, and

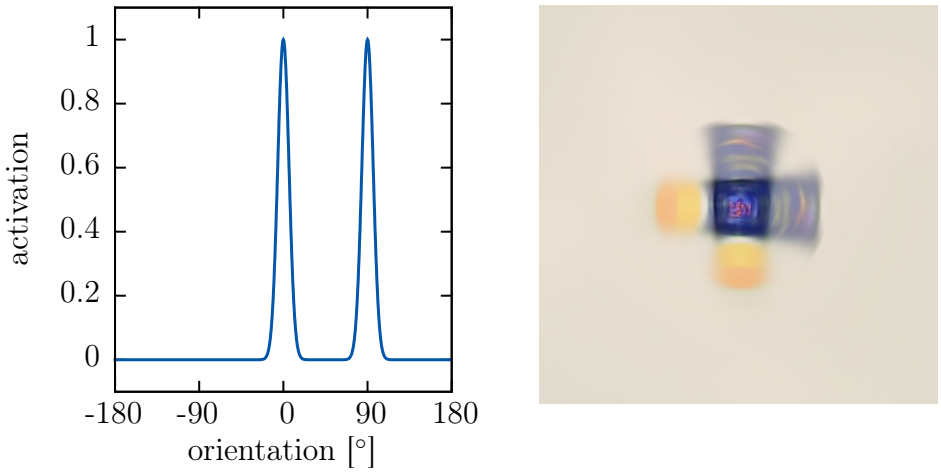


Figure 3.3: An illustration of the pose representation, here for orientation. Two orientation candidates are represented in the orientation estimate shown in the left panel. When applied to an image, they create a superposition, which is illustrated in the right panel.

---

compare this pattern with all learned views, W_l, where $l \in \{1, 2, \ldots\}$ is an arbitrarily chosen label for a view learned in the training procedure described later. For each label, the inner product yields a match value

$$\text{match}_l^{\text{sp}}(t) = \iint \hat{I}_{\text{bu}}^{\text{rot}}(x, y, t)\ \hat{W}_l(x, y, t)\ dx\ dy. \tag{3.3}$$

Here I introduce the notational convention that for a function, $F(\mathbf{x}, t)$, its normalized version (disregarding time) is indicated by a "hat" and calculated as

$$\hat{F}(\mathbf{x}, t) = \frac{F(\mathbf{x}, t) - \bar{F}(t)}{\|F(t)\|_2}, \tag{3.4}$$

where $\bar{F}(t)$ is the mean value of $F(\mathbf{x}, t)$, and $\|F(\mathbf{x}, t)\|_2$ indicates the L2-norm of $F(\mathbf{x}, t)$ (the time argument, t, is ignored in the calculation of both the mean and the L2-norm). $\text{match}_l^{\text{sp}}(t)$ corresponds to the match operations in the *identity* stage in Figure 3.1.

---

the patterns may be stored in the log-polar representation. However, inspection of the transformed and stored patterns is more convenient in Cartesian coordinates.

3.1.2 Matching pose in the top-down path

In the top-down path, the memorized views, $W_l(x, y, t)$, are superposed to yield

$$P_{\text{td}}(x, y, t) = \sum_l p_l(t)\ W_l(x, y, t), \tag{3.5}$$

where $p_l(t) \in [0, 1]$ (described in more detail in Equation 3.18) is the current identity estimate for the view which has been assigned the label, $l \in \{1, 2, \ldots\}$. This memory superposition corresponds to the transformation in the identity stage of the top-down path in Figure 3.1. The superposition is cross-correlated with the shifted bottom-up pattern to obtain a match value,

$$\text{match}^{\text{sp}}_{\text{rot}}(\phi, t) = \iint \hat{P}_{\text{td}}(\rho', \phi + \phi', t)\ \hat{I}^{\text{sh}}_{\text{bu}}(\rho', \phi', t)\ d\rho'\ d\phi'. \tag{3.6}$$

This match value serves as input to the neural fields that represent the orientation estimate (described in more detail in Section 3.2.1) and corresponds to the matching operation in the *orientation* stage in Figure 3.1 and 3.2.

To determine the shift between the memory representation and the input, the superposition is first transformed according to

$$P^{\text{rot}}_{\text{td}}(\rho, \phi, t) = \int p^{\text{inv}}_{\text{rot}}(\phi - \phi', t)\ P_{\text{td}}(\rho, \phi', t)\ d\phi'. \tag{3.7}$$

Here, $p^{\text{inv}}_{\text{rot}}(\phi - \phi', t) = p_{\text{rot}}(\phi' - \phi, t)$ is the inverse of the orientation estimate. $P^{\text{rot}}_{\text{td}}(\rho, \phi, t)$ corresponds to the transformation operation in the *orientation* stage in Figure 3.1. It is matched with the input to obtain

$$\text{match}^{\text{sp}}_{\text{sh}}(x, y, t) = \iint \hat{P}^{\text{rot}}_{\text{td}}(x + x', y + y', t)\ \hat{I}_{\text{sp}}(x', y', t)\ dx'\ dy', \tag{3.8}$$

which serves as input to the fields that represent the shift estimate (described in more detail in Section 3.2.1) and corresponds to the matching operation in the *position* stage in Figure 3.1.

3.2 Neural dynamics

Pose is represented by functions that assign for each pose parameter value how much that value is estimated to match the input. As an example, assume that an object is rotated by 45° to the correspond learned view. This could be represented by an orientation estimate with the shape of a Gaussian centered on 45°. The advantage of such a representation is that multiple pose

candidates may be encoded at the same time. For example, the representation could have the shape of the sum of two Gaussians, one at 45°, and one at 135°, to express pose uncertainty due to symmetries in the object.

This capacity to represent multiple candidates is crucial to the neural dynamics of the estimation process. Initially, the pose estimates are flat, meaning that all pose parameter values are estimated to match equally well. The resulting patterns in the top-down and bottom-up path are thus broad, but in accordance with the ordering property of superpositions (Arathorn, 2001, 2002), they can be used to refine the pose and identity estimates over time by gradually reducing the contribution of mismatching poses. Such a refinement is realized by the neural dynamics of the pose and identity representation described next.

3.2.1 Pose representation

All pose parameters are represented using the same principle. I therefore describe the neural representation of pose in terms of a generic pose parameter, which I denote by r. For translation, $r = (x, y)$, and for orientation, $r = \phi$.

The value of each estimated pose parameter is represented in two layers of dynamic neural fields, both governed by neural dynamics analogous to the dynamic neural field equation (Equation 2.1). For the first layer, this dynamics is

$$\begin{aligned}\tau_1 \dot{u}_1(r,t) = &- u_1(r,t) + h_1 + w_\xi \, \xi(r,t) \\ &+ s_1(r,t) \\ &+ \left[k^{u_1} * (g \circ u_1)\right](r,t)\,,\end{aligned} \tag{3.9}$$

with parameters defined analogous to Equation 2.1. Note that because this equation describes all pose fields, each variable would need an additional index to indicate the transformation parameter. However, to avoid clutter, I suppress these indices. The inputs, $s_1(r,t)$, are the match functions described above. That is, for translation,

$$s_1(x,y,t) = \left[k^{\mathrm{sh,sp}} * \mathrm{match}^{\mathrm{sp}}_{\mathrm{sh}}\right](x,y,t)\,, \tag{3.10}$$

where $k^{\mathrm{sh,sp}}$ defines an excitatory coupling, and for rotation,

$$s_1(\phi,t) = \left[k^{\mathrm{rot,sp}} * \mathrm{match}^{\mathrm{sp}}_{\mathrm{rot}}\right](\phi,t)\,, \tag{3.11}$$

where again $k^{\mathrm{rot,sp}}$ defines an excitatory coupling.

The activation of the fields on the second layer is governed by the dynamics

$$\begin{aligned}\tau_2 \dot{u}_2(r,t) = &- u_2(r,t) + h_2 + w_\xi\, \xi(r,t) \\ &+ w\,[G_\sigma * (\sigma_+ \circ u_1)\,](r,t) \\ &+ [k^{u_2} * (g \circ u_2)\,](r,t)\,,\end{aligned} \tag{3.12}$$

where the parameters are again defined analogous to Equation 2.1. The activation of the first layer is coupled into the second layer through a semi-linear transfer function,

$$\sigma_+(u) = \begin{cases} u: & u \geq 0 \\ 0: & \text{otherwise.} \end{cases} \tag{3.13}$$

The current estimate for the pose parameter value of the system is a multiplicative mixture of the two field layers:

$$p_T(r,t) = m(t)\,\sigma(u_2(r,t)) + (1 - m(t))\,\sigma_+(u_1(r,t))\,, \tag{3.14}$$

where T is an index for a transformation. The factor of the mixture is $m(t) = \sigma(u^{\mathrm{pd}}(t))$, the output of a peak-detector (see Equation 2.2) whose dynamics is given by

$$\begin{aligned}\tau \dot{u}^{\mathrm{pd}}(t) = &- u^{\mathrm{pd}}(t) + h + w_\xi\, \xi(t) \\ &+ \int k^{\mathrm{p},1} \sigma(u_1(r,t))\, dr \\ &+ w^{\mathrm{pd}}\, g\left(u^{\mathrm{pd}}(t)\right),\end{aligned} \tag{3.15}$$

with time scale, τ_p, resting level, h_p, and self-excitation, $w^{\mathrm{pd}} > 0$, and where the indices for the specific transformation have again been dropped. This dynamics is parameterized so that the activation, $u^{\mathrm{pd}}(t)$, grows above the threshold when a peak forms in the first layer of the pose representation.

The first layer has weak global inhibition (here denoted as γ_1 in analogy to the definition of the interaction kernel in Equation 2.3). Multiple candidates for the pose of the object may thus become active. Candidates that are strong enough to pierce the threshold are further strengthened by local excitation, while weak candidates are suppressed by global inhibition. The candidates from the first layer excite the second layer. However, global inhibition on the second layer is stronger (that is, $\gamma_2 > \gamma_1$), so that the strongest candidate is selected.

The timescales of the layers are constrained by $\tau_1 < \tau_2$, so that the first layer converges faster than the second layer. Thus, candidate poses initially

become active in the first layer. Because of the slower time scale of the second layer, it forms a peak much later than the first layer. Thus, the pose mixture (Equation 3.14) initially contains multiple candidate values which transform the input image and the superposition of the memorized views. Over time, this leads to increasingly accurate estimates for the object's pose. When the second layer becomes active, it makes a decision on this reduced set of candidates, selecting the strongest as the final estimated pose and stabilizing that decision against perturbations.

3.2.2 Identity representation

The system learns individual object views. Each view, W_l, is given an arbitrary label, $l \in \{1, 2, \ldots\}$, during training. In analogy to pose, identity is represented by two layers of dynamic neural nodes called *label nodes*. The activation, $u_{1,l}$, of the first-layer node for label l is governed by the dynamics

$$\begin{aligned}\tau_1 \dot{u}_{1,l}(t) = &- u_{1,l}(t) + h_1 + w_\xi \, \xi(t) \\ &+ s_{1,l}(t) - \gamma_1 \sum_{l' \neq l} g\left(u_{1,l'}(t)\right) \\ &+ w^{\mathrm{u}_1} \, g\left(u_{1,l}(t)\right), \end{aligned} \tag{3.16}$$

where the parameters τ_1, and h_1 are defined analogous to the parameters of the dynamic neural node equation (Equation 2.7), $s_{1,l}(t) = \mathrm{match}_l^{\mathrm{sp}}(t)$, w^{u_1} defines the self-excitation of the node, and $\gamma_1 > 0$ creates competition between all active label nodes on the first layer.

The activation of the nodes on the second layer is governed by the dynamics

$$\begin{aligned}\tau_2 \dot{u}_{2,l}(t) = &- u_{2,l}(t) + h_2 + w_\xi \, \xi(t) \\ &+ w^{2,1} \, \sigma_+\left(u_{1,l}(t)\right) - \gamma_2 \sum_{l' \neq l} g\left(u_{2,l'}(t)\right) \\ &+ w^{\mathrm{u}_2} \, g\left(u_{2,l}(t)\right), \end{aligned} \tag{3.17}$$

where the parameters τ_2 and h_2 are again defined analogous to the parameters of Equation 2.7, w^{u_2} defines the self-excitation, and $\gamma_2 > 0$ creates competition between all active label nodes on the second layer.

The current estimate for label l is again a multiplicative mixture of these two layers of nodes:

$$p_l(t) = m(t)\, \sigma(u_{2,l}(t)) + (1 - m(t))\, \sigma_+\left(u_{1,l}(t)\right), \tag{3.18}$$

where the factor of the mixture is $m(t) = \sigma(u^{\mathrm{pd}}(t))$, with $u^{\mathrm{pd}}(t)$, the activation of a node that detects the presence of an active label node on the second layer with dynamics analogous to a peak detector,

$$\begin{aligned}\tau \dot{u}^{\mathrm{pd}}(t) = &- u^{\mathrm{pd}}(t) + h + w_{\xi}\ \xi(t) \\ &+ \sum_{l} k^{\mathrm{p},1} \sigma(u_{1,l}(t)) \\ &+ w^{\mathrm{pd}}\ g\left(u^{\mathrm{pd}}(t)\right), \end{aligned} \tag{3.19}$$

with time scale, τ, resting level, h, and self-excitation, $k^{\mathrm{p},1} > 0$.

As with the pose representation, the nodes on the first layer evolve on a faster timescale than the second layer ($\tau_1 < \tau_2$), and the selectivity in the second layer is greater than in the first one ($\gamma_2 > \gamma_1$) so that only a single node may be active on the second layer. This winning node represents the label recognized by the system and, ultimately, the final recognition decision of the system.

3.3 Localized color and edge orientation histograms

In the previous section, I explained the architecture for spatial patterns because they serve as a good illustration of the principles. However, the concrete extraction of the spatial pattern in the original work (Faubel and Schöner, 2009, 2010; Lomp et al., submitted 2016) relies on relatively coarse spatial patterns. These patterns have low discriminative power and mainly serve segmentation purposes. The main discriminative power of the architecture comes from its other feature channels, which use locally extracted histograms of color (hue from the HSV color model) and edge orientations (extracted by applying steerable filters (Freeman and Adelson, 1991) separately to the luma, Y, and chroma, Cr and Cb, channels of the image) as features. This approach yields five different match values for pose and labels (one for the color histograms, three for the different edge histograms, and one for the spatial pattern channel). These match values are all input to the fields and nodes representing pose and labels. They are combined by weighted addition, with the weights chosen to reflect the discriminative power of each feature channel.

3.3.1 Histogram extraction

The localized histograms for a feature channel, F, are extracted from a feature image, $I_F(x, y)$ (this could, for example, be the responses from an edge orientation filter). For computational efficiency, histograms are extracted on a regular two-dimensional grid that samples the image space at positions $\mathbf{c}_{i,j}$, where i and j index positions on the grid. Histograms are extracted according to

$$h^F_{i,j}(f,t) = n_F(t) \iint G_{\sigma_h,\mathbf{c}_{i,j}}(x,y)\, m_F(x,y,t)\, \chi_{[f,f+\Delta f]}\left(I_F(x,y,t)\right)\, dx\, dy. \tag{3.20}$$

Here, f is a feature value (for example, color), $G_{\sigma_h,\mathbf{c}_{i,j}}$ is a Gaussian with width σ_h, centered on $\mathbf{c}_{i,j}$:

$$G_{\sigma,\mathbf{c}}(\mathbf{x}) = G_\sigma(\mathbf{x}-\mathbf{c}) = \exp\left(-\frac{(\mathbf{x}-\mathbf{c})^2}{2\sigma^2}\right), \tag{3.21}$$

and $\chi_{[f_1,f_2]}$ is the characteristic function,

$$\chi_{[f_1,f_2]}(f) = \begin{cases} 1: & f \in [f_1,f_2] \\ 0: & \text{otherwise,} \end{cases} \tag{3.22}$$

with a sampling interval Δf that reflects the discrete sampling of the feature space in the implementation. The masking term,

$$m_F(x,y,t) = \theta_F(x,y,t)\; P^{\text{sh}}_{\text{td}}(x,y,t), \tag{3.23}$$

is determined by the predicted top-down pattern from the spatial channel,

$$P^{\text{sh}}_{\text{td}}(x,y,t) = \iint p_{\text{sh}}(x'-x, y'-y, t)\; P^{\text{rot}}_{\text{td}}(x',y',t)\; dx'\, dy', \tag{3.24}$$

and another mask $\theta_F(x,y,t) \in \{0,1\}$ obtained by applying a threshold to the input. For the color channel, this is

$$\theta_{\text{col}}(x,y,t) = \begin{cases} 1: & \text{saturation}(x,y,t) > \theta_{\text{sat}} \\ & \wedge\ \text{value}(x,y,t) > \theta_{\text{val}} \\ 0: & \text{otherwise} \end{cases}, \tag{3.25}$$

where $\text{saturation}(x,y,t)$ is the saturation channel from the HSV color model, and $\text{value}(x,y,t)$ is the value channel of the HSV color model, and $\theta_{\text{sat}}, \theta_{\text{val}} \in \mathbb{R}$ are thresholds. For the edge channels, the threshold is

$$\theta_{\text{edge}}(x,y,t) = \begin{cases} 1: & \text{energy}(x,y,t) > \theta_{\text{energy}} \\ 0: & \text{otherwise} \end{cases}, \tag{3.26}$$

where energy(x, y, t) is the energy of the output of steerable filters (Freeman and Adelson, 1991) and, again, $\theta_{\text{energy}} \in \mathbb{R}$ is a threshold. Finally, the normalization term is given by

$$n_F(t) = \left(\iint \theta_F(x, y, t) \; dx \; dy \right)^{-1}. \tag{3.27}$$

3.3.2 Translating histograms

The goal is to match memorized histograms with the current input. The current shift estimate defines which of the localized histograms extracted from the input contribute to this matching. To achieve this, the input histograms of the feature dimension, F (sampled by f) are mapped to a single bottom-up histogram, weighted by the shift representation:

$$h_{\text{bu}}^{F,\text{sh}}(f, t) = \sum_{i,j} p_{\text{sh}}^{\max}(\mathbf{c}_{i,j}, t) \; h_{i,j}^{F}(f, t). \tag{3.28}$$

Because the input space is subsampled, the shift estimate must be subsampled as well. This is expressed by $p_{\text{sh}}^{\max}(\mathbf{c}_{i,j}, t)$, which describes the maximum value of the shift estimate in a rectangular region around the histogram center, $\mathbf{c}_{i,j}$. The rectangular regions are non-overlapping and correspond to a region covering the sampling grid so that together, they cover the entire range of possible shift estimates. Sampling the shift estimate using the maximum better preserves the locations of peaks that might otherwise be lost in the subsampled estimate due to the coarseness of the sampling.

To determine the match value for the shift parameter, the top-down histogram, $h_{\text{td}}^{F}(f, t)$ is correlated with the localized histograms in the input according to

$$\text{match}_{\text{sh}}^{F}(\mathbf{c}_{i,j}, t) = \int \hat{h}_{i,j}^{F}(f', t) \hat{h}_{\text{td}}^{F}(f', t) \; df', \tag{3.29}$$

where the accent ("hat") again indicates mean-freeing and normalization (see Equation 3.4). As before, this match value serves as input to the dynamics for pose representation. However, due to the subsampling by the histogram centers, $\mathbf{c}_{i,j}$, $\text{match}_{\text{sh}}^{F}(\mathbf{c}_{i,j}, t)$ first has to be lifted back to full image sampling before being input into the position representation. In implementations, this is achieved by bicubic interpolation.

For the color channel ($F = \text{col}$), no further transformations are necessary, and $h_{\text{td}}^{\text{col}}(f, t)$ corresponds to the weighted superposition of the histograms memorized during training (denoted by $h_{\text{mem},l}^{\text{col}}$):

$$h_{\text{td}}^{\text{col}}(f, t) = \sum_{l} p_l \; h_{\text{mem},l}^{\text{col}}(f, t), \tag{3.30}$$

where, as before, l indexes an object view and p_l is the current label estimate (see Equation 3.18).

For the edge feature channels, the superposition must be rotated before it can be used for matching. Thus, the top-down histogram is given by $h_{\text{td}}^{\text{edge}} = h_{\text{td}}^{\text{edge,rot}}$, the rotated superposition of the memorized histograms, described in the next section.

3.3.3 Rotating histograms

Due to the use of isotropic Gaussian weights for the extraction of the color histograms, they are invariant under rotation around the Gaussian's center. The edge channels, however, vary with rotation. In the present section, I describe how the current rotation estimate can be applied to edge histograms. In the full system, there are three edge channels (see Section 3.3), however, I write down the equations for a single edge channel as the equations are analogous for all channels.

In the bottom-up path, the shifted edge histogram is rotated by transforming along the orientation dimension:

$$h_{\text{bu}}^{\text{edge,rot}}(\theta, t) = \int p_{\text{rot}}(\theta - \theta', t)\; h_{\text{bu}}^{\text{edge,sh}}(\theta', t)\; d\theta', \tag{3.31}$$

where p_{rot} is the current rotation estimate (see Equation 3.14). In the top-down path, the weighted superposition of the histograms memorized during training is transformed according to

$$h_{\text{td}}^{\text{edge,rot}} = \int p_{\text{rot}}^{\text{inv}}(\theta - \theta', t)\; h_{\text{td}}^{\text{edge}}(\theta', t)\; d\theta', \tag{3.32}$$

where $p_{\text{rot}}^{\text{inv}}$ is the inverse of the rotation estimate, and $h_{\text{td}}^{\text{edge}}$, in analogy to Equation 3.30, is calculated as the weighted superposition of the memorized patterns, $h_{\text{mem},l}^{\text{edge}}$,

$$h_{\text{td}}^{\text{edge}}(\theta, t) = \sum_l p_l\; h_{\text{mem},l}^{\text{edge}}(\theta, t). \tag{3.33}$$

The localized edge histograms contribute to the estimation of shift as described in Section 3.3.2. They are also sensitive to the orientation of the object and thus contribute to its estimation by correlation with the shifted bottom-up pattern,

$$\text{match}_{\text{rot}}^{\text{edge}}(\theta, t) = \int h_{\text{bu}}^{\text{edge,sh}}(\theta + \theta', t)\; h_{\text{td}}^{\text{edge}}(\theta', t)\; d\theta'. \tag{3.34}$$

This match function contributes to the dynamic neural fields for representing the object's orientation (Equation 3.9).

3.3.4 Matching histograms

Object identity is estimated by matching the transformed bottom-up histograms, h^F_{bu}, to the learned patterns, $h^F_{\text{mem},l}$ (where l again indexes the object views). Similar to the spatial channel, this match is obtained with the mean-free normalized correlation, here along the feature dimension,

$$\text{match}^F_l(t) = \int \hat{h}^F_{\text{bu}}(f,t)\ \hat{h}^F_{\text{mem},l}(f,t)\ df. \tag{3.35}$$

For the color channel ($F = \text{col}$), $f = c$ is a color, and the shifted bottom-up histogram is used for matching ($h^{\text{col}}_{\text{bu}}(c,t) = h^{\text{col,sh}}_{\text{bu}}(c,t)$). For the edge channels ($F = \text{edge}$), $f = \theta$ is an edge orientation, and the shifted and rotated histogram is used for matching ($h^{\text{edge}}_{\text{bu}}(\theta,t) = h^{\text{edge,rot}}_{\text{bu}}(\theta,t)$).

3.4 Fusing feature channels

The match values for object identity and pose from the five feature channels are fused by providing additive input to the first-layer identity and pose representations. Different channels contribute to different dimensions of the pose estimate, as illustrated in Figure 3.4. Formally, the input to the first layer pose fields (cf. Equation 3.9) is thus

$$s_1(r,t) = \sum_F \left[k^{\text{P,F}} * \text{match}^F_P\right](r,t)\,, \tag{3.36}$$

where $k^{\text{P,F}} \in \mathbb{R}$ is an excitatory coupling kernel for the feature channel $F \in \{\text{sp, col, Y edge, Cr edge, Cb edge}\}$. The pose parameter is indicated by $P \in \{\text{sh, rot, sc}\}$, and match^F_T is one of the match functions described above.

Analogously, the input to the first layer label nodes (cf. Equation 3.16) is

$$s_{1,l}(t) = \sum_F k^{\text{L,F}}\ \text{match}^F_l(t), \tag{3.37}$$

with weight, $k^{\text{L,F}} \in \mathbb{R}$, and F and defined as above.

Because edges have no polarity, the edge channels provide orientation estimates in the range 0° to 180°, whereas the shape channel provides orientation estimates up to 360°. In principle, shape could thus disambiguate the orientation estimate from the edge channels. However, the shape channel is usually coarse and thus has low discriminance. Therefore this is not done, and the rotation estimate is instead restricted to the smaller range.

3.5 Learning object views

The architecture learns object views in a supervised training phase. During training, the time-continuous dynamics of the system are simulated while objects are presented one by one. For each image, the system first undergoes an externally triggered reset in which the resting level of all fields and nodes is lowered. This leads to a decay of residual activation from previous trials. After the reset, the resting levels are restored, and the supervisor provides label and pose information to the system. This is necessary because the system cannot properly match the pose and label of the object before learning it. The supervisor biases the first layer pose estimates by adding a Gaussian input centered around values corresponding to identity transformations. These inputs are strong enough to induce a peak at the cued location which then suppresses, via global inhibition, any other peaks that may form due to matches with objects that are already known. Note that the induced peaks have a non-zero width, and only the means of these peaks correspond to an identity transform. As a result, the learned pattern is a slightly blurred version of the input.

Analogously, an arbitrary label is provided by the supervisor in the form of an additive bias to a label node on the first layer. The cued label node thus becomes active and excites the corresponding node on the second layer. The strong global inhibition on the second layer suppresses all other nodes, ensuring that only the cued label is represented.

The pattern for a label, l, is learned for each feature channel, F, in a pattern memory, $W_l^F(\mathbf{f}, t)$. These pattern memories adapt based on the dynamics

$$\tau_{\text{mem}} \dot{W}_l^F(\mathbf{f}, t) = -\left(W_l^F(\mathbf{f}, t) - P^F(\mathbf{f}, t)\right)\ b_{\text{lrn}}(t)\ p_l(t). \tag{3.38}$$

Through the linear term, $-\left(W_l^F(\mathbf{f}, t) - P^F(\mathbf{f}, t)\right)$, the dynamics converges to the feature pattern, $P^F(\mathbf{f}, t)$ with timescale τ_{mem}. The factor $b_{\text{lrn}}(t) \in \{0, 1\}$ enables and disables learning and is controlled externally. After the second layer converges to the cued label, the current label parameter estimate, $p_l(t)$ (see Equation 3.18), is one for the cued label and zero everywhere else. The modulation by this value ensures that only the pattern memory of the cued label changes.

The pattern to be memorized, $P^F(\mathbf{f}, t)$, is the fully transformed pattern from the bottom-up path. For the spatial channel, it is given by $P^{\text{sp}}(x, y, t) = I_{\text{bu}}^{\text{rot}}(x, y, t)$. For the color histogram channel, $P^{\text{col}}(c, t) = h_{\text{bu}}^{\text{col,sh}}(c, t)$. For the edge channels, $P^{\text{edge}}(\theta, t) = h_{\text{bu}}^{\text{edge,rot}}(\theta, t)$.

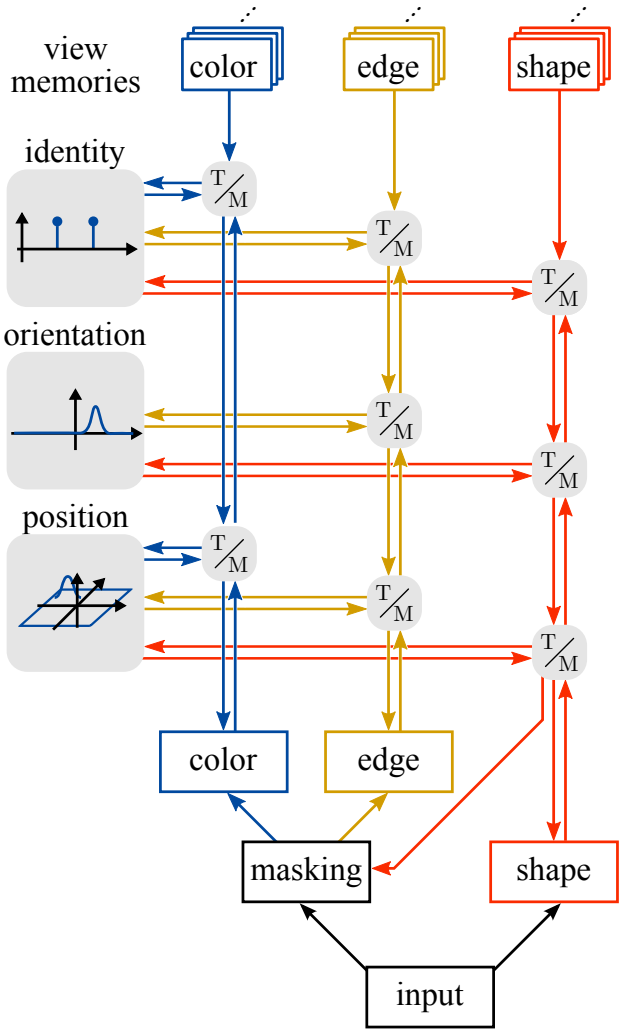


Figure 3.4: Connections between the different feature channels and pose parameters. Boxes labeled "T/M" indicate transformation and matching of the patterns.

Chapter 4

Scene representation

In the second part of my thesis, I investigate how attention may guide the recognition of multiple objects in an input image that contains multiple objects, and how the recognized identities may be committed to a working memory representation. This is based on a scene representation model (Zibner et al., 2011b; Zibner and Faubel, 2015), which I describe in the present chapter. This model may, for instance, extract the color and size for each salient location in a scene (that is, an image containing one or more objects) and store this information in a space-color working memory and a space-size working memory field. Once a representation is formed for the objects in the scene, the architecture offers ways to query the representation, indicating the location of cued features values, or the values memorized for attended locations (though I omit these querying capacities in the description because I do not use them in my model).

In the present chapter, I describe the dynamics of the scene representation architecture for an exemplary color channel. Additional feature channels merely require duplicating the structures specific to the feature channel and do not require changes to the underlying framework.

4.1 Saliency

The goal of the saliency computation in the scene representation architecture is to indicate locations of the input image, $I(x, y)$, that are part of the foreground based on heuristics extrapolated from the human vision system. Saliency is computed by first entering feature information from the input image into an *early space-color field* (shown in the middle of Figure 4.1). The activation of this field is given by $u^{\text{espc}}(x, y, c, t)$, where (x, y) is a location in the input space, c is a color, and t is the time argument. The activation

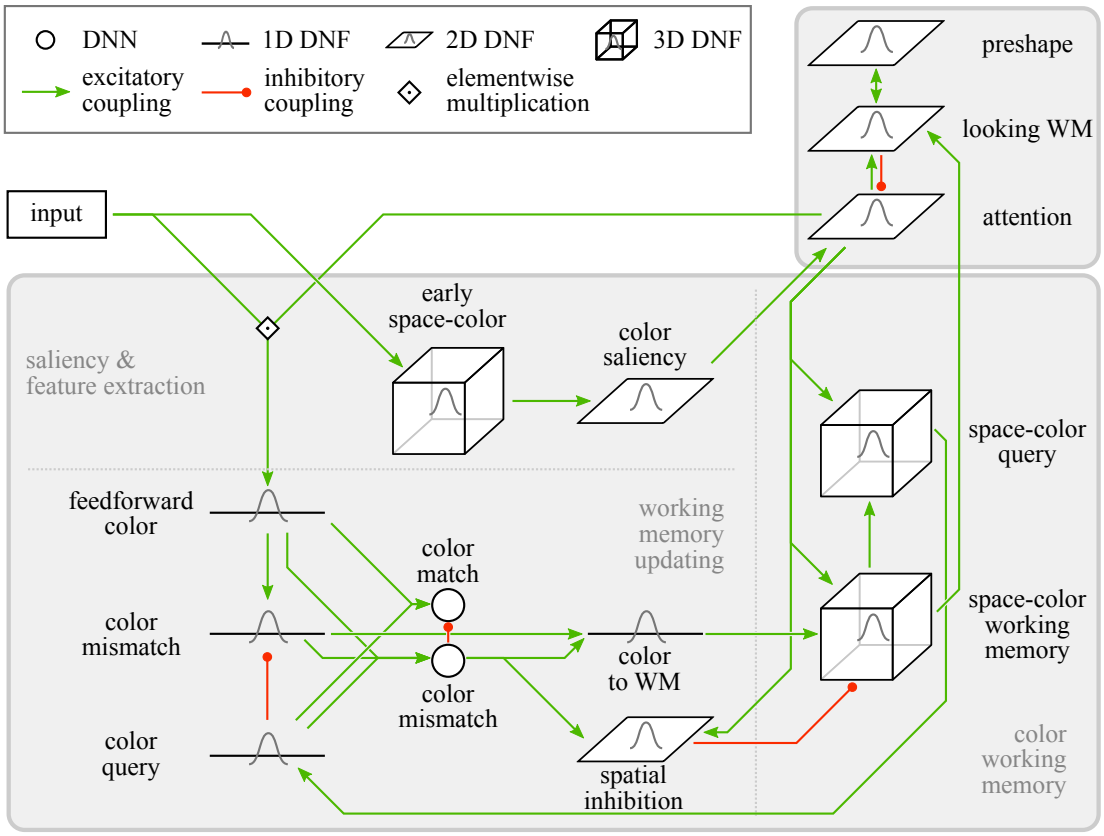


Figure 4.1: An overview over the scene representation architecture.

---

follows the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{espc}}(x,y,c,t) = &- u^{\text{espc}}(x,y,c,t) + h + w_\xi\ \xi(x,y,c,t) \\ &+ \left[k^{\text{espc,I}} * s^{\text{espc}}\right](x,y,c,t) \\ &+ \left[k^{\text{espc}} * (g \circ u^{\text{espc}})\right](x,y,c,t)\,. \end{aligned} \tag{4.1}$$

The field's input derives from the image as

$$s^{\text{espc}}(x,y,c,t) \;=\; \begin{cases} S\,(x,y,t): & H\,(x,y,t) = c \\ 0: & \text{otherwise,} \end{cases} \tag{4.2}$$

where $H\,(x,y,t)$ and $S\,(x,y,t)$ are the hue and saturation values (from the HSV color model) at the location x, y of the input image. The field's interaction kernel is defined so that similar colors compete when they are spatially distant, whereas they excite each other when they are spatially close. Formally, the interaction is

$$k^{\text{espc}}(\Delta x, \Delta y, \Delta c) = w_{\text{exc}}\ G_{\sigma_d}\,(\Delta x, \Delta y, \Delta c) - w_{\text{inh}}\ G_{\sigma_c}\,(\Delta c)\,, \tag{4.3}$$

where $w_{\text{exc}} > w_{\text{inh}} > 0$, G_σ is a zero-mean Gaussian with width σ, and $\sigma_c, \sigma_d > 0$. Through the inhibition of similar colors, colors that occur frequently tend to induce a lower activation level in the early space-color field than those that occur less frequently, leading to pop-out effects.

The saliency information in the early space-color field is contracted (see Chapter 2) to the two spatial dimensions and convolved to obtain a color saliency map,

$$s^{\text{sal}}_{\text{col}}(x,y,t) = \iiint G_{\sigma_S}(x-x', y-y')\, g\left(u^{\text{espc}}(x',y',c,t)\right)\, dx'\, dy'\, dc. \quad (4.4)$$

This color saliency map serves as input to a color saliency field (also shown in the middle of Figure 4.1) whose dynamics is

$$\begin{aligned}\tau \dot{u}^{\text{csal}}(x,y,t) = &- u^{\text{csal}}(x,y,t) + h + w_\xi\, \xi(x,y,t) \\ &+ s^{\text{sal}}_{\text{col}}(x,y,t) \\ &+ \left[k^{\text{csal}} * \left(g \circ u^{\text{csal}}\right)\right](x,y,t)\,. \end{aligned} \quad (4.5)$$

The field suppresses weak local maxima through global inhibition so that locations with low saliency are inhibited. It also strengthens boundaries between salient locations through local inhibition, leading to a better distinction between them.

4.2 The attention field and the looking working memory

The attention field (top right corner of Figure 4.1) is not specific to any feature channel, but instead receives additive input from each channel's saliency field. From this combination, it determines which location should be attended by selecting the most salient location using global inhibition. Its dynamics are

$$\begin{aligned}\tau \dot{u}^{\text{atn}}(x,y,t) = &- u^{\text{atn}}(x,y,t) + h + w_\xi\, \xi(x,y,t) \\ &+ w_1 \left[G_{\sigma_1} * \left(g \circ u^{\text{csal}}\right)\right](x,y,t) \\ &- w_2 \left[G_{\sigma_2} * \left(g \circ u^{\text{lwm}}\right)\right](x,y,t) \\ &+ w_3 \iiint G_{\sigma_3}(x-x', y-y')\, u^{\text{scq}}(x',y',c,t)\, dx'\, dy'\, dc \\ &+ s^{\text{atn}}_{\text{bhv}}(t) \\ &+ \left[k^{\text{atn}} * \left(g \circ u^{\text{atn}}\right)\right](x,y,t)\,, \end{aligned} \quad (4.6)$$

where u^{scq} is the activation of the space-color query field described in Equation 4.12, and $s^{\text{atn}}_{\text{bhv}}$ is an input from behavioral organization that is described in Section 4.4. u^{lwm} (*looking WM* in the top-right corner of Figure 4.1) is a field that memorizes which locations were recently inspected and inhibits these locations in the attention field. This inhibition prevents the attention field from selecting the same salient location again immediately after it was inspected and is at the core of the sequential scanning of salient locations. The dynamics of the looking working memory is

$$\begin{aligned}\tau \dot{u}^{\text{lwm}}(x,y,t) = &- u^{\text{lwm}}(x,y,t) + h + w_\xi\ \xi(x,y,t) \\ &+ w_1 \left[G_{\sigma_1} * p^{\text{lwm}}\right](x,y,t) \\ &+ \iiint G_{\sigma_2}(x-x', y-y')\ u^{\text{scm}}(x',y',c,t)\ dx'\ dy'\ dc \\ &+ \left[k^{\text{lwm}} * \left(g \circ u^{\text{lwm}}\right)\right](x,y,t)\,, \end{aligned} \tag{4.7}$$

where the triple integral is the contracted and convolved output of the space-color working memory field ($u^{\text{scm}}(x,y,c,t)$, see Section 4.3) and $s^{\text{lwm}}_{\text{bhv}}$ is an input from the behavioral organization described in Section 4.4. $p^{\text{lwm}}(x,y,t)$ (see *preshape* in the top-right corner of Figure 4.1) preshapes the looking working memory field so that only locations with sufficiently strong input from the preshape may form working memory peaks. The preshape's dynamics follows that described in Zibner et al. (2011b),

$$\begin{aligned}\dot{p}^{\text{lwm}}(x,y,t) = &\frac{g\left(u^{\text{atn}}(x,y,t)\right)}{\tau_{\text{buildup}}}\left(-p^{\text{lwm}}(x,y,t) + g\left(u^{\text{atn}}(x,y,t)\right)\right) \\ &+ \frac{1 - g\left(u^{\text{atn}}(x,y,t)\right)}{\tau_{\text{decay}}}\left(-p^{\text{lwm}}(x,y,t).\right) \end{aligned} \tag{4.8}$$

This dynamics consists of two terms. The first term with timescale τ_{buildup} builds up activation at locations which are above threshold in the attention field. The second term lets activation decay at locations at which the attention field is below threshold. This decay happens on a slower timescale, $\tau_{\text{decay}} > \tau_{\text{buildup}}$. Through this preshape, the attended location is pushed into the working memory regime so that self-sustained peaks may form. When the attention field focuses on another object, preshape decays until a critical level at which the looking working memory locations leave the working memory regime. Peaks in the looking working memory thus decay, and the locations may be attended again.

4.3 Space-feature fields and working memory

To memorize the feature value at the attended location, it is first read out by a *feedforward color* field (lower left corner of Figure 4.1), whose dynamics reads

$$\begin{aligned}\tau \dot{u}^{\mathrm{ffc}}(c,t) = &- u^{\mathrm{ffc}}(c,t) + h + w_\xi \; \xi(c,t) \\ &+ G_\sigma \left(c - c^{\mathrm{ffc}}(t)\right) + g\left(u^{\mathrm{apd}}(t)\right) \\ &+ \left[k^{\mathrm{ffc}} * \left(g \circ u^{\mathrm{ffc}}\right)\right](c,t)\,, \end{aligned} \tag{4.9}$$

where u^{apd} is the activation of a peak detector for the attention field (cf. Equation 2.2), and

$$c^{\mathrm{ffc}}(t) = \max \; H\left(x,y,t\right) \; g\left(u^{\mathrm{atn}}(x,y,t)\right), \tag{4.10}$$

with $H\left(x,y,t\right)$, the current hue of the input image at location (x,y) according to the HSV color model.

The *color query* field (also shown in the lower left corner of Figure 4.1) reads out the color for the attended location from working memory using the dynamics

$$\begin{aligned}\tau \dot{u}^{\mathrm{cq}}(c,t) = &- u^{\mathrm{cq}}(c,t) + h + w_\xi \; \xi(c,t) \\ &+ w \left[G_\sigma * \left(g \circ u^{\mathrm{scq}}\right)\right](x,y,c,t) \\ &+ \left[k^{\mathrm{cq}} * \left(g \circ u^{\mathrm{cq}}\right)\right](c,t)\,. \end{aligned} \tag{4.11}$$

Its input is the contracted activation from the *space-color query* field (right side of Figure 4.1), which combines attention (expanded to a tube input over color) with the output of the space-color working memory (denoted by u^{scm}, see below),

$$\begin{aligned}\tau \dot{u}^{\mathrm{scq}}(x,y,c,t) = &- u^{\mathrm{scq}}(x,y,c,t) + h + w_\xi \; \xi(x,y,c,t) \\ &+ w_1 \left[G_{\sigma_1} * \left(g \circ u^{\mathrm{atn}}\right)\right](x,y,t) \\ &+ w_2 \left[G_{\sigma_2} * \left(g \circ u^{\mathrm{scm}}\right)\right](x,y,c,t) \\ &+ \left[k^{\mathrm{scq}} * \left(g \circ u^{\mathrm{scq}}\right)\right](x,y,c,t)\,, \end{aligned} \tag{4.12}$$

The *color mismatch field* (again shown in the lower left corner of Figure 4.1) receives input from both the color query field and feedforward color field:

$$\begin{aligned}\tau \dot{u}^{\mathrm{cm}}(c,t) = &- u^{\mathrm{cm}}(c,t) + h + w_\xi \; \xi(c,t) \\ &+ w_{\mathrm{m}} \left[G_{\sigma_{\mathrm{m}}} * \left(g \circ u^{\mathrm{ffc}}\right)\right](c,t) + k^{\mathrm{cm,cmi}} \; g\left(u^{\mathrm{cmi}}\right) \\ &- w_{\mathrm{m}} \left[G_{\sigma_{\mathrm{m}}} * \left(g \circ u^{\mathrm{cq}}\right)\right](c,t) \\ &+ \left[k^{\mathrm{cm}} * \left(g \circ u^{\mathrm{cm}}\right)\right](c,t)\,, \end{aligned} \tag{4.13}$$

where u^{cmi} is the activation of a color mismatch node described in Equation 4.20. Together, the color mismatch field, color query field and feedforward color field implement the structure of the three-layer change detection model presented by Johnson et al. (2009). Input from the feedforward color field is canceled out by the input from working memory (read out via the color query field) if both represent the same color. The color mismatch field therefore only forms a peak either when there is no color stored in working memory (it is excited by the feedforward input, which induces a peak), or when the stored color does not match the color in the input (the excitation from the feedforward input is not canceled out by the working memory contents).

The mismatch information is input to the *color to working memory* field (*color to WM* in Figure 4.1):

$$\begin{aligned}\tau \dot{u}^{\text{c2wm}}(c,t) = &- u^{\text{c2wm}}(c,t) + h + w_\xi \, \xi(c,t) \\ &+ w \left[G_\sigma * (g \circ u^{\text{cm}}) \right](c,t) \\ &+ \left[k^{\text{c2wm}} * \left(g \circ u^{\text{c2wm}} \right) \right](c,t) \,, \end{aligned} \tag{4.14}$$

which, together with the attention field, is input to the *space-color working memory* (right-hand side of Figure 4.1):

$$\begin{aligned}\tau \dot{u}^{\text{scm}}(x,y,c,t) = &- u^{\text{scm}}(x,y,c,t) + h + w_\xi \, \xi(x,y,c,t) \\ &+ w_1 \left[G_{\sigma_1} * \left(g \circ u^{\text{c2wm}} \right) \right](c,t) \\ &+ w_2 \left[G_{\sigma_2} * \left(g \circ u^{\text{atn}} \right) \right](x,y,t) \\ &- w_3 \left[G_{\sigma_3} * \left(g \circ u^{\text{spin}} \right) \right](x,y,t) \\ &+ \left[k^{\text{scm}} * (g \circ u^{\text{scm}}) \right](x,y,c,t) \,. \end{aligned} \tag{4.15}$$

The *spatial inhibition field* (bottom of Figure 4.1), $u^{\text{spin}}(x,y,t)$, inhibits the attended location across all colors and thus ensures that any previously stored color information is deleted from working memory when new information is written. Its dynamics is given by

$$\begin{aligned}\tau \dot{u}^{\text{spin}}(x,y,t) = &- u^{\text{spin}}(x,y,t) + h + w_\xi \, \xi(x,y,t) \\ &+ w_2 \left[G_{\sigma_2} * \left(g \circ u^{\text{atn}} \right) \right](x,y,t) \\ &+ s^{\text{spin}}_{\text{bhv}}(t) \\ &+ \left[k^{\text{spin}} * \left(g \circ u^{\text{spin}} \right) \right](x,y,t) \,. \end{aligned} \tag{4.16}$$

The input from the behavioral organization, $s^{\text{spin}}_{\text{bhv}}$ (see Section 4.4), ensures that the field only forms a peak when there is a mismatch between the stored and perceived color.

4.4 Sequential scanning through behavioral organization

A task input (see Section 2.3) called *explore* (indexed by exp) enables the sequential scanning of salient locations. This scanning is mediated by the behaviors *explore scene* and *inspect object*.

The intention node of the *explore scene* behavior has the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{int}}_{\text{exps}}(t) = &- u^{\text{int}}_{\text{exps}}(t) + h + w_\xi\ \xi(t) \\ &+ s^{\text{task}}_{\text{exp}}(t) \\ &+ w^{\text{int}}_{\text{exps}}\ g\left(u^{\text{int}}_{\text{exps}}(t)\right)\end{aligned} \tag{4.17}$$

A condition of satisfaction is not formalized for this behavior. The dynamics of the intention node of the *inspect object* behavior is

$$\begin{aligned}\tau \dot{u}^{\text{int}}_{\text{ins}}(t) = &- u^{\text{int}}_{\text{ins}}(t) + h + w_\xi\ \xi(t) \\ &+ s^{\text{task}}_{\text{exp}}(t) + w^{\text{ins,exps}}\ g\left(u^{\text{int}}_{\text{exps}}(t)\right) - w^{\text{ins,cma}}\ g\left(u^{\text{int}}_{\text{cma}}(t)\right) \\ &+ w^{\text{int}}_{\text{ins}}\ g\left(u^{\text{int}}_{\text{ins}}(t)\right).\end{aligned} \tag{4.18}$$

The intention node is inhibited by the activation of a color match node, $u^{\text{int}}_{\text{cma}}(t)$. This node is part of the three-layer match structure that signals when the feature values of the currently attended locations have been committed to working memory successfully (more details on this are given below). It functions as the condition of satisfaction node of the inspect object attention.

When the intention to inspect an object activates, it boosts the attention field by providing input $s^{\text{atn}}_{\text{bhv}}(t) = w^{\text{atn,ins}}\ u^{\text{int}}_{\text{ins}}(t)$. This causes the attention field to select an object based on salience and the memory of previously inspected locations. Subsequently, the selected object is inspected, that is, its color is extracted and matched to the working memory contents by the three layer match structure. The fields involved in the change detection (color query field, feedforward color field and color match field) are described in Section 4.3. For their behavioral organization, two additional nodes are necessary that detect whether the fields signal a match or mismatch (or neither). The first is the *color match node*, which signals that the color stored in working memory and the input match. Its activation is governed by the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{cma}}(t) = &- u^{\text{cma}}(t) + h + w_\xi\ \xi(t) \\ &+ g\left(u^{\text{pd}}_{\text{ffc}}(t)\right) + g\left(u^{\text{pd}}_{\text{cq}}(t)\right) - w^{\text{cma,cmi}}\ g\left(u^{\text{cmi}}(t)\right) \\ &+ w^{\text{cma}}\ g\left(u^{\text{cma}}(t)\right),\end{aligned} \tag{4.19}$$

where $u^{\text{pd}}_{\text{ffc}}(t)$ is the activation of a peak detector for the feedforward color field, and $u^{\text{pd}}_{\text{cq}}(t)$ is a peak detector for the color query field. The second is the *color mismatch node*, which signals that the color stored in working memory and the input do not match. Its activation is governed by the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{cmi}}(t) = &- u^{\text{cmi}}(t) + h + w_\xi \; \xi(t) \\ &+ g\left(u^{\text{pd}}_{\text{ffc}}(t)\right) + g\left(u^{\text{pd}}_{\text{cq}}(t)\right) + g\left(u^{\text{pd}}_{\text{cm}}(t)\right) \\ &+ w^{\text{cmi}} \; g\left(u^{\text{cmi}}(t)\right), \end{aligned} \tag{4.20}$$

where $u^{\text{pd}}_{\text{cm}}(t)$ is the activation of a peak detector for the color mismatch field.

When the color in working memory and input match, the peak in the color query and feedforward color field have a similar shape, so that the effective input to the color mismatch field is close to zero (see above). The mismatch field is therefore the only of the three fields that does not form a peak. The peaks in the query and feedforward fields excite both the match and mismatch node. However, the mismatch node is parameterized so that it may only become active when all three fields form a peak. For the match node, these two peaks are sufficient for it to become active, and thus a match is signaled.

When input color and memorized color are misaligned, a peak is also formed in the mismatch field, and the mismatch node becomes active. Because its timescale is chosen to be slightly faster than that of the match node, the inhibitory coupling to the match node takes effect before the its activation reaches threshold. This inhibition is chosen to be so strong that suprathreshold activation of the match node is not possible when the mismatch node is active.

In a third case, there may not be any color information in working memory for the attended location, for example, because the location is inspected for the first time. In this case, a peak forms in the feedforward field and the mismatch field, and both the match and mismatch node remain inactive. Importantly, this means that no match is signaled, leading to the currently perceived color being committed to working memory before the next location is attended (see below).

When a mismatch is detected, this leads to an update of the feature values stored in working memory. The mismatch node excites the spatial inhibition field, $u^{\text{spin}}(x, y)$, by providing input $s^{\text{spin}}_{\text{bhv}}(t) = g\left(u^{\text{cmi}}(t)\right)$, and thus destabilizes any previously memorized color information for the attended location. The mismatch field also forms a peak at the color of the input (this also happens when no working memory entry is present because the color query field is empty and thus does not suppress the peak in the mismatch

field). This peak raises activation around the input color in the space-color working memory field through a slice input. A peak forms at the locations at which the slice overlaps with the attention, and the new color is thus committed to memory. The color query field reflects this update by forming a peak at the newly committed color and thus suppresses the peak in the mismatch field. This destabilizes the mismatch node, releasing inhibition of the match node, which subsequently may become active.

The match node provides the condition of satisfaction for the inspect object behavior. Its intention therefore becomes inactive, and the boost to the attention field is deactivated. Without attention, the peak in the feedforward color field dissolves, and the match node deactivates. The inspect object intention is no longer suppressed, and the next object may be selected, starting the process anew.

Chapter 5

Biologically inspired multiscale keypoints

In the third part of my thesis, I use edge orientation histograms to describe local image regions. Extracting these features is computationally costly, and it is therefore desirable to do so only at regions of the image that are highly informative. Rodrigues and du Buf (2009) describe a method for finding points in an image that identify such regions. These points, referred to as *keypoints*, are extracted from the maxima of complex cell responses based on mechanisms inspired by the responses of orientation-sensitive simple and complex cells found in visual cortex (Hubel and Wiesel, 1962, 1968). They can be calculated fast and have been shown to reduce the amount of data necessary for image description (Terzić et al., 2013).

In the first stage of keypoint extraction, the image, $I(x, y)$, is convolved with a set of Gabor wavelets at different orientations, θ, and wavelengths, λ (which correspond to a scale, $\sigma = 0.56\,\lambda$, see Terzić et al. 2013). The wavelets are described by

$$g_{\lambda,\theta}\,(x, y) = \exp\left(-\frac{\tilde{x}^2 + 0.5\tilde{y}^2}{2\sigma^2}\right)\ \exp\left(i\frac{2\pi\tilde{x}}{\lambda}\right) \tag{5.1}$$

where

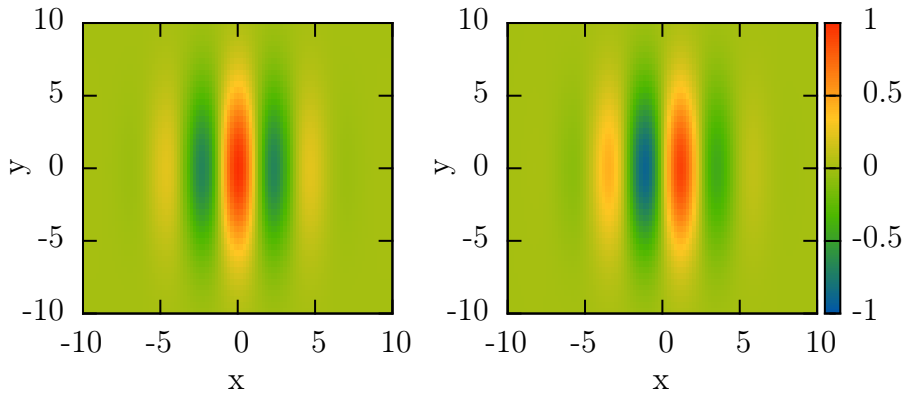
$$\tilde{x} = x\cos\,(\theta) + y\sin\,(\theta) \tag{5.2}$$

and

$$\tilde{y} = y\cos\,(\theta) - x\sin\,(\theta)\,. \tag{5.3}$$

Convolving the image with this filter yields the simple cell responses,

$$R_{\lambda,\theta}(x, y) = [I * g_{\lambda,\theta}]\,(x, y)\,. \tag{5.4}$$



(a) Real part (even wavelet). (b) Imaginary part (odd wavelet).

Figure 5.1: The real and imaginary part of a Gabor wavelet with orientation, $\theta = 0$, and wavelength, $\lambda = 5$.

The complex cell responses are given by the magnitude of the simple cell responses,

$$C_{\lambda,\theta}(x,y) = |R_{\lambda,\theta}(x,y)| = \sqrt{\mathrm{Re}\left(R_{\lambda,\theta}(x,y)\right)^2 + \mathrm{Im}\left(R_{\lambda,\theta}(x,y)\right)^2}. \tag{5.5}$$

Due to the complex exponent in the second factor of the Gabor wavelet equation, the real and imaginary parts of this equation represent responses of filters that are phase-shifted versions of each other. The real part corresponds to an *even* wavelet with its maximal value aligned with its main axis (see Figure 5.1(a)), whereas the imaginary part corresponds to an *odd* wavelet with its maximal value offset from its main axis (see Figure 5.1(b)). The even wavelet thus responds maximally to lines that fall directly onto the center of the receptive field, whereas the odd wavelet responds maximally to edges.

Line crossings at finer scales may lead to gaps in the responses of the even and odd wavelets. Similarly, line terminations may be difficult to locate precisely because responses decay smoothly past them (Rodrigues and du Buf, 2009). To address these issues, an additional layer of processing is added in which the complex cell responses are modified based on a set of inhibitory kernels. These kernels are defined relative to the main axes of the Gabor filters, using $d_s = 0.6\lambda \sin(\theta)$ and $d_c = 0.6\lambda \cos(\theta)$.

In this additional layer, double stopped kernels as well as tangential and

radial inhibition are applied. The double stopped kernels are given by

$$k^{\mathrm{D}}_{\lambda,\theta}(x,y) = \delta(x,y) - \frac{1}{2}\cdot\left(\delta(x-2d_s, y+2d_c) + \delta(x+2d_s, y-2d_c)\right), \quad (5.6)$$

where $\delta(x,y)$ is the delta function (one when $x=0$ and $y=0$, zero otherwise). The tangential inhibition kernel is given by

$$k^{\mathrm{TI}}_{\lambda,\theta}(x,y) = -2\delta(x,y) + \delta(x+d_c, y+d_s) + \delta(x-d_c, y-d_s). \quad (5.7)$$

Finally, the radial inhibition kernel which is effective across orientations (see Equation 5.9) is given by

$$k^{\mathrm{RI}}_{\lambda,\theta}(x,y) = \delta\left(x+\frac{d_c}{2}, y+\frac{d_s}{2}\right) + \delta\left(x-\frac{d_c}{2}, y-\frac{d_s}{2}\right). \quad (5.8)$$

With these kernels, the final keypoint activation map for a wavelength, λ, is calculated as

$$\begin{aligned} K_\lambda(x,y) = &\sum_{\theta=0}^{\pi} \sigma_+\left(\left[C_{\lambda,\theta} * k^{\mathrm{D}}_{\lambda,\theta}\right](x,y)\right) \\ &- \sum_{\theta'=0}^{2\pi} \sigma_+\Big(\left[C_{\lambda,\theta'} * k^{\mathrm{TI}}_{\lambda,\theta'}\right](x,y) \\ &\quad + \left[C_{\lambda,\theta'+\frac{\pi}{2}} * k^{\mathrm{RI}}_{\lambda,\theta'}\right](x,y) - C_{\lambda,\theta'}(x,y)\Big). \end{aligned} \quad (5.9)$$

Keypoints are the local maxima of these keypoint activation maps and may be found by thresholding the activation and then fitting parabolas onto the local maxima (Terzić et al., 2013).

Chapter 6

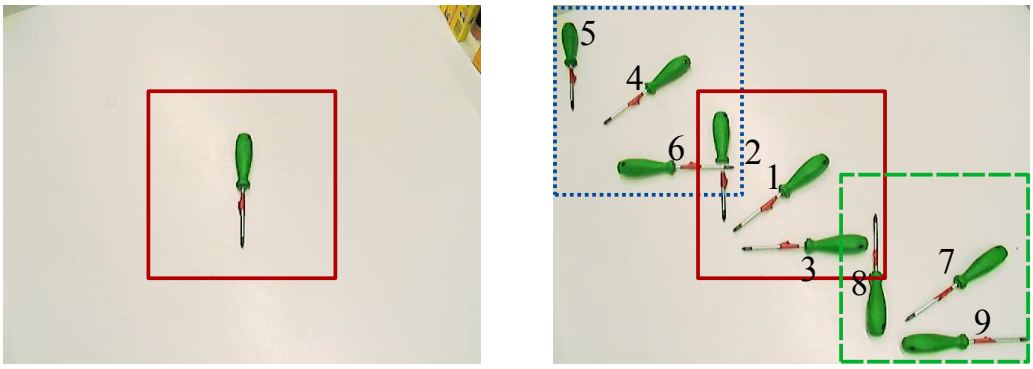
Image databases

To quantitatively evaluate the object recognition systems I describe and to compare their performance, I rely on four databases which I describe in the present chapter. One is a tabletop dataset from prior work on object recognition in dynamic field theory (Faubel and Schöner, 2008, 2009, 2010; Lomp et al., submitted 2016). The second is a variant of this database that specifically tests the recovery of in-plane transformations. The third dataset, the COIL-100 dataset (Nene et al., 1996), is an established dataset that investigates performance for depth rotations. The fourth dataset is a custom database which consists of images containing multiple objects. In the next sections, I describe these databases in more detail.

6.1 The tabletop database

The tabletop database[1] was initially developed to test the label-feature field based object recognition system (Chapter 3, Faubel and Schöner, 2008), and a variant was later used to evaluate the histogram-based object recognition system described in Chapter 3 (Faubel and Schöner, 2009, 2010; Lomp et al., submitted 2016). The dataset contains images of thirty common household objects (shown in Figure 6.2). Each image shows a single object in one of ten different poses on a white tabletop in front of the robotic platform CoRA (Iossifidis et al., 2003). Of the ten poses, one is a training pose which shows the object in the center of the image with its elongated axis aligned with the vertical axis of the image plane (see Figure 6.1(a)). The other nine poses, shown in Figure 6.1(b), are used for testing recognition performance.

[1]The database is available online at `https://www.ini.rub.de/pages/publications/LompFaubelSchoener2016`.



(a) Training pose. (b) Test poses.

Figure 6.1: Training and test poses for the tabletop dataset. Numbers indicate different test poses. Rectangles indicate the region used as input. In the test set, the region indicated by the dotted blue rectangle is used for poses 4–6, the region indicated by the solid red rectangle is used for poses 1–3, and the region indicated by the dashed green rectangle is used for poses 7–9. The image showing the test poses is synthesized from multiple test images.

Images were captured using one of CoRA's Sony DFW-VL500 color cameras at a resolution of 640×480 pixels. Lighting conditions as well as camera parameters were kept constant for the acquisition of the database.

To evaluate recognition performance, a subregion of 256×256 pixels is cut out from the test images. This subregion provides the input to the object recognition architecture. It reflects attentive processes that would focus the input onto the vicinity of an object (much like the one described in Part II). The placement of the region depends on the pose of the object in the image; for poses 1–3, the region is placed in the center of the image. For poses 4–6, it is placed at the top-left corner. For poses 7–9, it is placed at the lower right corner. Figure 6.1 illustrates these regions for an exemplary object.

It is worth noting that even with the cutting-out, objects in this database take up only a small portion of the input image. For example, the pencil sharpener, one of the smaller objects, covers an area of roughly 40×60 pixels in the training image. Furthermore, the elevated position of the camera along with the way that objects are placed on the table mean that the objects are often seen from a single side, and that the different poses of the objects often correspond approximately to transformations in the image plane. Also, the uniformly white background simplifies the problem of segmenting foreground and background.

Figure 6.2: Training views of the thirty objects of the tabletop database.

6.2 The transformed database

In the *transformed* database, the training images are the same as in the tabletop database. The test images, however, are generated by algorithmically transforming the training images by applying, in the following order, scaling by a factor from the interval $[0.8, 1.2]$, rotations of up to 360°, and translations in the range of $[-50, 50]$ pixels separately in the x and y direction. The exact transformation applied to the images is selected by sampling uniform random distributions over the specified intervals, and test images are generated online, so that in theory, infinitely many test images may be obtained, sampling part of the continuous range of parameter values of the pose transformations estimated by the object recognition system.

6.3 The COIL database

The Columbia object image library (COIL-100; Nene et al., 1996) comprises color images of 100 different objects. During image acquisition, the objects were placed on a rotating plate in front of a uniformly black background. Color images were taken at 5° intervals, so that a total of 72 images per object are available, showing the objects from different viewing angles sampled from the horizontal viewing plane.

In accordance with previous work on object recognition in dynamic field theory (Faubel and Schöner, 2009; Lomp et al., submitted 2016), I use only the first thirty objects of the COIL database (see Figure 6.3 for the zero-degree views of these objects). In addition, because the images were originally recorded at a resolution of 128×128 pixels, I pad them to twice that resolution by adding a border of 64 pixels to achieve a format similar to the tabletop database. This border has a uniform color sampled from the background of the original images to avoid artifacts at the borders.

6.4 Multi-object database

One of the architectures I propose is designed specifically to recognize multiple objects in a single image. In order to evaluate the performance of the system, I created a database containing pictures of twenty everyday objects in a setting analogous to the tabletop database presented above. The database is available online.[2] Pictures were taken with the camera of the robotic platform Caren (the successor to the CoRA platform) and therefore

[2] https://www.ini.rub.de/pages/publications/LompPhDThesis

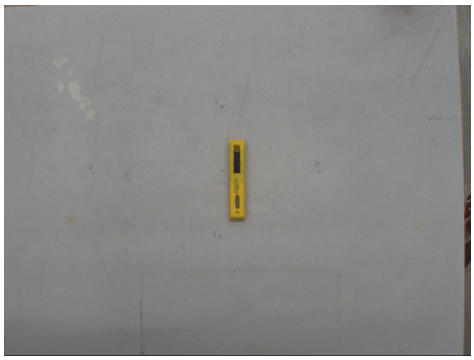
Figure 6.3: Training views of the thirty objects from the COIL-100 database.

Figure 6.4: Single-object training views of all twenty objects of the multi-object database. Images are cropped to show only the relevant part.

---

have a viewing angle that is relatively close to a bird's-eye view. Effects of viewing angle were further rectified by inverting perspective effects with estimated parameters of the camera transformation in order to align the pixel coordinate system with an arbitrarily chosen coordinate frame on the table. As in the tabletop database, the background is a relatively uniformly white table, and camera settings as well as illumination were kept constant for the acquisition of the images. Images are captured in color, using a Sony XCD-SX90CR camera at a resolution of 800×600 pixels.

The database contains two types of training data that allow for different training methods. In the *single object training* set, objects are presented individually in a standard pose (placed in the center of the image, with their long axis pointing up as in the tabletop dataset). Figure 6.5(a) shows examples of full training images. Figure 6.4 shows the training images of



(a) Training image containing a single object.

(b) Training image containing multiple objects.

Figure 6.5: Examples for the types of training images of the multi-object database.

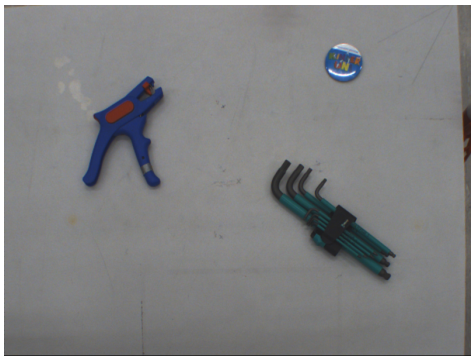
---

all twenty objects, cropped to remove excess background. In the *multiple object training set*, sets of three objects are arbitrarily paired into training images (the last training image contains only two objects), for a total of seven training images which show each objects exactly one time. An example training image from this set is shown in Figure 6.5(b).

The database has a total of 102 test images. Each image contains one, two, three or seven objects, which I again paired arbitrarily. Figure 6.6 shows examples for these images. Each test image is also assigned to either the *close* or *spaced* category to indicate the spatial arrangement of the objects (see bottom row of Figure 6.6 for examples). I annotated the pose of all objects in the test set by manually aligning single-object training images overlaid on top of the objects in the test image.

(a) Image containing one object.

(b) Image containing two objects.



(c) Image containing three objects.

(d) Image containing seven objects, large separation (*spaced*)

(e) Image containing seven objects, small separation (*close*)

Figure 6.6: Examples of different types of test pictures in the multi-object database.

Part II

Attention and working memory for object recognition

Introduction

Introspection tells us that our visual perception of the world is instantaneous and without gaps. When we look at a picture, we feel that we immediately see and recognize all objects in it. However, there is much more sequentiality in our perception than we are aware of (see, for example, Simons and Levin, 1997; Simons and Chabris, 1999). This sequentiality is likely due to the overwhelming amount of information that enters through our eyes at any given moment (Koch et al., 2006). Processing all of this data at once would exceed the capacities available to our brain. Instead, we sequentially focus attention on individual locations in the image, shifting focus either overtly by moving our eyes, or covertly without any physical movement (Posner, 1980). How such sequentiality may arise from the continuous-time dynamics of the nervous system is an open question.

Though the object recognition system presented by Faubel and Schöner (2009, 2010) and Lomp et al. (submitted 2016, see also Chapter 3) is not intended as a model of such sequential attention, some analogies to attentional mechanisms in the brain can be found. When the system has estimated the position of an object, the features at this position are brought into the focus of the object recognition system, which uses them as the basis for identification. In that sense, the system attends this recognized object. However, the architecture only ever attends a single object. Without external control, it would attend the recognized object forever. Furthermore, when external control does trigger a new recognition, the old recognition is forgotten immediately because there is no mechanism in the system that forms a memory of past recognitions. Questions about sequential recognition of objects are therefore not answered by this system alone.

The scene representation architecture presented in Chapter 4, on the other hand, models sequential processing of the input using mechanisms inspired by processing of visual information in the brain. However, the object descriptions that the model uses rely on basic features such as color, and object identity is not part of the model.

Both systems thus solve complementary aspects of the sequential pro-

cessing of visual input. Integrating them with each other is therefore the main topic of the present part. This integration is supported by modularity inherent to dynamic field theory architectures (Lomp et al., accepted 2016). However, to make full use of this modularity, both architectures need to be formulated entirely within dynamic field theory. This is not true for the object recognition architecture as described in Chapter 3, because transitioning from one object to the next requires external control, and because the input region of the image must still be provided externally.

I therefore begin the present part with a chapter that describes how the recognition process may be realized within the framework of dynamic field theory. In the chapter following this description, I make use of the fully neuro-dynamic object recognition architecture and integrate it with the scene representation architecture. I finish the present part with an evaluation of the combined architecture, as well as a discussion of the results and a comparison to similar approaches in the literature.[3]

[3]Please note that the work presented here is based on initial implementations presented in the Master's thesis of Philipp Hebing, which I supervised.

Chapter 7

Behavioral organization of object recognition

The transition between recognition of different objects in the object recognition system presented by Faubel and Schöner (2009, 2010) and Lomp et al. (submitted 2016) is solved externally, using algorithmic processes outside of the framework of dynamic field theory. To address this, two problems must be solved. First, the system must detect that a new recognition should be started. I realize this with a set of transient detectors that detect changes in the input image and trigger a new recognition when the magnitude of these changes exceeds a threshold. Second, when the transient detectors trigger a new recognition, the object recognition architecture may still represent the previous recognition decision. A new recognition started in this state would be biased strongly towards the object recognized due to the stabilization mechanisms inherent to the label nodes and pose fields. Thus, a mechanism is needed that overcomes this stabilization. In the experiments on the original architecture (Faubel and Schöner, 2009, 2010; Lomp et al., submitted 2016), this is achieved by an externally activated inhibitory boost, which resets all label nodes and pose fields. To achieve a similar effect without external control, behavioral organization (see Section 2.3) is required. I therefore introduce a new behavior *recognize*. When this behavior is inactive, the label nodes and pose fields relax to their negative resting levels, which correspond to the inhibitory boost of the external reset. Because this makes the activation of the nodes and fields fall below threshold, potential biases due to self-stabilization are removed when this behavior is inactive. Once the behavior does become active, the intention node of the *recognize* behavior boosts the fields and nodes back to a resting level that allows recognition to take place.

The *reset* behavior precedes this behavior and ensures that a reset takes

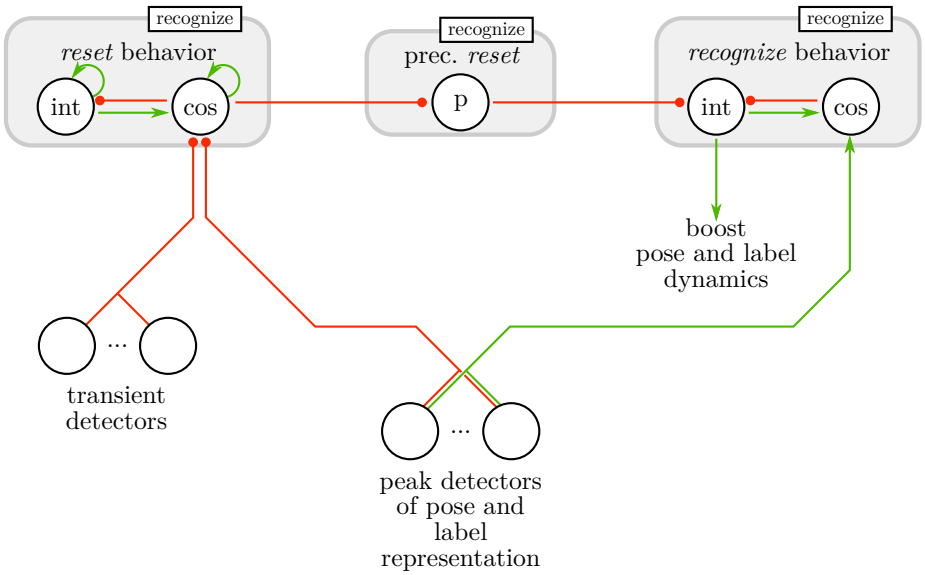


Figure 7.1: Connectivity of the behaviors involved in recognition.

place every time the input changes. Change of the input is indicated by a set of transient detectors. Because their signal is only above threshold for a short duration, the behavior's intention node forms a memory of the signal, which is erased when its condition of satisfaction is reached. This is the case when activation in the layer two label nodes and pose fields falls below threshold and all biases have been removed.

The decay of activation in the second layers of the pose and label fields, in turn, is triggered because the condition of satisfaction of the *reset* behavior is a precondition for the intention node of the *recognize* behavior (see Section 2.3). Reactivation of the behavior is prevented because the behavior's condition of satisfaction node is in memory mode.

Over the next sections, I formalize this behavioral organization.

7.1 The *reset* behavior

Following the scheme described in Berger et al. (2012), the transient detectors that drive the *reset* behavior (see left side of Figure 7.1) consist of two layers of neural fields each. One layer is an *inhibitory layer* with the dynamics

$$\tau_{\text{tri}} \dot{u}_C^{\text{tri}}(x, y, t) = -u_C^{\text{tri}}(x, y, t) + s_C(x, y, t), \tag{7.1}$$

where x, y are image coordinates and $s_C(x, y, t)$ is the input from an image channel C (either the hue, saturation or value channel of the HSV color

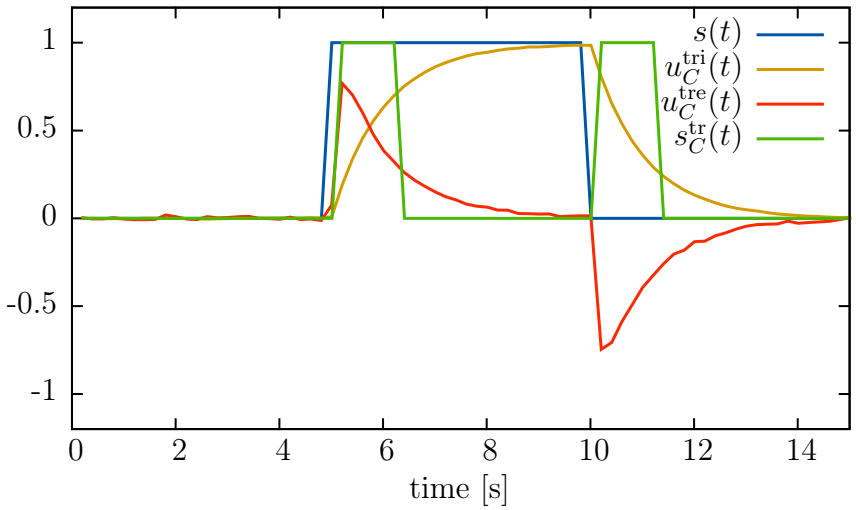


Figure 7.2: An exemplary time course of the signals involved in transient detection. For the illustration, the involved fields and signals have been reduced to a single point by omitting the spatial dimension. The transient signal, $s_C^{\mathrm{tr}}(t)$, briefly becomes active when the input, $s(t)$, changes its value.

---

space). This layer inhibits the *excitatory layer*, whose dynamics is given by

$$\tau_{\mathrm{tre}}\dot{u}_C^{\mathrm{tre}}(x,y,t) = -u_C^{\mathrm{tre}}(x,y,t) + s_C(x,y,t). - u_C^{\mathrm{tri}}(x,y,t) \tag{7.2}$$

The timescales of the layers are constrained by $\tau_{\mathrm{tre}} < \tau_{\mathrm{tri}}$. As a result, the activation of the excitatory layer becomes non-zero for a brief period of time whenever the input signal changes (see Figure 7.2). The sign of the activation of the excitatory layer depends on the change in the input (if the value of $s_C^{\mathrm{tri}}(x,y,t)$ increases, the sign is positive, if it decreases, the sign is negative). To get the final transient signal for a channel, I use the absolute value of the contracted activation of the excitatory layer,

$$s_C^{\mathrm{tr}}(t) = H\left(\iint \left|u_C^{\mathrm{tre}}(x,y,t)\right| \; dx \; dy - \theta\right), \tag{7.3}$$

with the Heaviside function, $H(\cdot)$. A transient detector thus activates when the change of the input exceeds a threshold, θ. An exemplary time course of $s_C^{\mathrm{tr}}(t)$ and its terms is shown in Figure 7.2.

Recognition is enabled by the task *recognize*. The associated task input,

$s^{\text{task}}_{\text{rcgn}}(t)$, is input to the intention node of the *reset* behavior:

$$\begin{aligned}\tau \dot{u}^{\text{int}}_{\text{res}}(t) = &- u^{\text{int}}_{\text{res}}(t) + h + w_{\xi}\ \xi(t) \\ &+ w_1\ s^{\text{task}}_{\text{rcgn}}(t) - w_2\ g\left(u^{\text{cos}}_{\text{res}}(t)\right) \\ &+ w^{\text{int}}_{\text{res}}\ g\left(u^{\text{int}}_{\text{res}}(t)\right). \end{aligned} \tag{7.4}$$

The *reset* behavior's condition of satisfaction node inhibits the intention node, and thus the intention is deactivated when the condition of satisfaction is achieved. The dynamics of the condition of satisfaction node is

$$\begin{aligned}\tau \dot{u}^{\text{cos}}_{\text{res}}(t) = &- u^{\text{cos}}_{\text{res}}(t) + h + w_{\xi}\ \xi(t) \\ &+ w_1\ s^{\text{task}}_{\text{rcgn}}(t) + w_2\ g\left(u^{\text{int}}_{\text{res}}(t)\right) \\ &- w_3 \sum_{C} s^{\text{tr}}_{C}(t) - w_4 \sum_{T} g\left(u^{\text{pd}}_{T}(t)\right) \\ &+ w^{\text{cos}}_{\text{res}}\ g\left(u^{\text{cos}}_{\text{res}}(t)\right). \end{aligned} \tag{7.5}$$

The node may only become active when the output of all peak detectors of the pose and label fields, $g(u^{\text{pd}}_{T}(t))$, is close to zero (T is again the set of pose parameters estimated by the object recognition system, for example, position, orientation and label, and $u^{\text{pd}}_{T}(t)$ is the activation of the corresponding layer two peak detectors; see Equation 3.15 on page 36). Thus, the condition of satisfaction (and hence the reset of the object recognition system) is achieved whenever peaks in the pose and label fields have decayed (this decay is triggered because the fields and nodes must be boosted by the *recognize* behavior to be able to form peaks; this is described in detail below). The condition of satisfaction node is in memory mode and therefore remains active once the reset is completed. Inhibition from the transient detectors' outputs $s^{\text{tr}}_{C}(t)$ destabilizes this memory, causing the condition of satisfaction node to deactivate whenever the input image changes sufficiently. Because the transient signal decays after a short time, inhibition also decays, and the condition of satisfaction may once again activate when the label nodes and pose fields are reset.

7.2 The *recognize* behavior

Recognition may begin when the *reset* behavior described above is complete, that is, when its condition of satisfaction is achieved. This dependency is

expressed by a precondition node (see center of Figure 7.1),

$$\begin{aligned}\tau \dot{u}^{\text{pre}}_{res}(t) = &- u^{\text{pre}}_{res}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{rcgn}}(t) - w_2 \; g\left(u^{\text{cos}}_{\text{res}}(t)\right) \\ &+ w^{\text{pre}}_{res} \; g\left(u^{\text{pre}}_{res}(t)\right), \end{aligned} \tag{7.6}$$

that is shown in the center of Figure 7.1. The intention node of the *recognize* behavior, shown on the right side of Figure 7.1, follows the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{int}}_{\text{rcgn}}(t) = &- u^{\text{int}}_{\text{rcgn}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{rcgn}}(t) - w_2 \; g\left(u^{\text{cos}}_{\text{rcgn}}(t)\right) - w_3 \; g\left(u^{\text{pre}}_{\text{res}}(t)\right) \\ &+ w^{\text{int}}_{\text{rcgn}} \; g\left(u^{\text{int}}_{\text{rcgn}}(t)\right), \end{aligned} \tag{7.7}$$

where $u^{\text{cos}}_{\text{rcgn}}(t)$ is the activation of the condition of satisfaction node of the *recognize* behavior (described in more detail below). The intention to recognize induces detection instabilities in the pose fields and label nodes of the object recognition architecture. This is achieved by choosing the resting levels for the fields and nodes to only allow the input from the match functions (see Section 3.2) to push activation of the labels and nodes above threshold when it is combined with the input from the *recognize* behavior's intention node.

Recognition is considered complete when peaks are formed on the second layers of the pose representations, and when a second-layer label node is active at the same time. This is expressed by the condition of satisfaction node of the *recognize* behavior (right side of Figure 7.1),

$$\begin{aligned}\tau \dot{u}^{\text{cos}}_{\text{rcgn}}(t) = &- u^{\text{cos}}_{\text{rcgn}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{rcgn}}(t) + w_2 \; g\left(u^{\text{int}}_{\text{rcgn}}(t)\right) + w_3 \sum_T g\left(u^{\text{pd}}_T(t)\right) \\ &+ w^{\text{cos}}_{\text{rcgn}} \; g\left(u^{\text{cos}}_{\text{rcgn}}(t)\right). \end{aligned} \tag{7.8}$$

Here, T is again the set of pose parameters estimated by the object recognition system, and $u^{\text{pd}}_T(t)$ is the activation of the corresponding layer two peak detectors (see Equation 3.15). The weight, w_3, for the input from these peak detectors is chosen to only allow the condition of satisfaction node to become active when all peak detectors are active while the inputs from the task and intention node are present as well.

Chapter 8

Integration with the scene representation architecture

In the previous chapter, I describe how externally controlled processes in the object recognition architecture in the original formulation (Faubel and Schöner, 2009, 2010; Lomp et al., submitted 2016) may be controlled using principles of behavioral organization. In the present chapter, I describe how the process of recognizing objects may be integrated into the scene representation model described in Chapter 4. This involves a description of the representation used for binding labels to space in working memory, and includes a description of how the behavioral organization of the two architectures is combined. I also describe how the process of learning object views may be organized to reduce the required amount of external input so that users only need to provide a label on request.

8.1 Space-feature fields for labels

The object recognition architecture constitutes another feature channel for the scene representation model. In this channel, labels are bound to spatial locations, using dynamics analogous to the feature-specific fields in the color channel. However, instantiating these dynamics is not trivial because the feature to be stored—the recognized label—is encoded in individual label nodes rather than a field defined over a continuous feature dimension such as color. The one-dimensional fields of the label-feature channel must therefore be replaced by sets of nodes as well. These nodes, however, are governed by the same dynamics as the label nodes (self-excitation and global inhibition), and these nodes are parameterized to fulfill the same role as their counterparts in the color channel. For example, the *label mismatch nodes* are analogous to

the color mismatch field. Therefore, a node in the label mismatch field may become active only if the label in the working memory representation does not match the one at the currently attended location.

Another exception are the three-dimensional space-feature fields (working memory and the feature query field). I realize these as a set of two-dimensional fields defined over space. Each of these fields represents locations at which a specific label l is present. The dynamics of the space-label working memory fields are given by

$$\begin{aligned}\tau \dot{u}_l^{\text{slwm}}(x,y,t) = &- u_l^{\text{slwm}}(x,y,t) + h \\ &+ \text{inputs} \\ &+ \left[k_{\text{intra}}^{\text{slwm}} * \left(g \circ u_l^{\text{slwm}}\right)\right](x,y,t) \\ &+ \sum_{l' \neq l} \left(\left[k_{\text{inter}}^{\text{slwm}} * \left(g \circ u_{l'}^{\text{slwm}}\right)\right](x,y,t)\right), \end{aligned} \tag{8.1}$$

where 'inputs' is a shorthand for connectivity analogous to that of the space-color working memory field defined in Equation 4.15. Interaction between these space-label working memory fields is defined by two kernels: $k_{\text{intra}}^{\text{slwm}}$ realizes interaction of the field with itself, while $k_{\text{inter}}^{\text{slwm}}$ realizes interaction between fields for different labels. Taken together, the space-label fields for all labels effectively act like a three-dimensional space-label field with a kernel that is discrete in one dimension and can thus be thought of as analogous to the space-feature working memory fields.

Extending the scene representation architecture presented in Chapter 4 by a label channel is straightforward (connectivity and equations remain analogous, merely the indices of the variables and functions change). Space-label working memory fields are introduced together with space-label query fields. Similarly, label match and mismatch fields are introduced to provide input to the space-label working memory fields. However, the feedforward feature field does not receive input from a bank of filters, as this is not possible for the labels. Instead, it receives input from the second layer of the label nodes (again, these are considered here to be a field defined over a discrete feature dimension). On its own, this input is not strong enough to yield suprathreshold activation. Only when coupled with a global boost from the condition of satisfaction node of the *recognize* behavior (described in Section 7.2) does one of the label nodes form suprathreshold activation. This ensures that the label information is only committed to working memory when the recognition process is complete. The unavailability of a feedforward path for the labels also means that labels do not contribute to saliency computation.

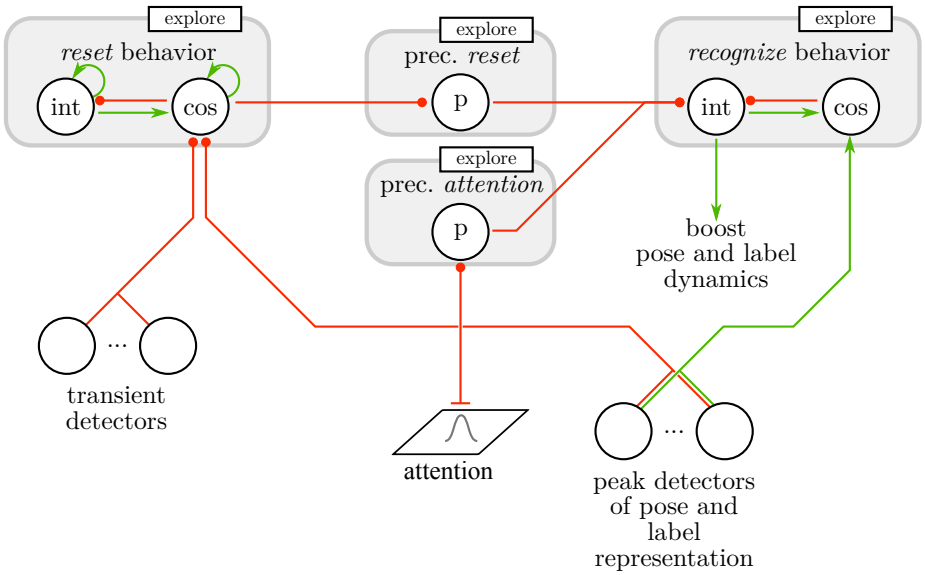


Figure 8.1: Connectivity of the behaviors involved in recognition in conjunction with the scene representation.

8.2 Guiding object recognition based on attention

Besides the addition of the label-feature channel to the scene representation architecture, the integration of the two architectures requires further modifications of the principles of object recognition and scene representation architectures described in Chapter 7 and Chapter 4, respectively. The scene representation's *explore* task replaces the *recognize* task of the modified object recognition approach from Chapter 7. Additionally, the center of the region of interest of the object recognition system is determined by the attended location,[1]

$$I^C_{\text{attn}}(x, y, t) = \frac{\iint g\left(u^{\text{atn}}(x + x', y + y', t)\right) I(x', y', t)\ dx'\ dy'.}{\iint g\left(u^{\text{atn}}(x'', y'', t)\right)\ dx''\ dy''.}, \tag{8.2}$$

[1] In the implementation, this convolution is approximated to increase efficiency. The location of the highest activation in the attention field is determined, and a subregion of the image corresponding to that location is extracted programmatically. This also includes border handling, which extends the values at the borders for regions not covered by the image.

where C is again one of the channels of the input image (hue, saturation or value). Note that for simplicity of notation, I assume that the center of the input image, I, is located at the origin. The result of Equation 8.2 serves as input to the object recognition system.

This attentional control gives rise to another constraint for the behavioral organization: recognition is only meaningful when a location is attended, that is, when a peak is present in the attention field. I therefore introduce a precondition *attention* (see Figure 8.1), whose precondition node is governed by the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{pre}}_{attn}(t) = &- u^{\text{pre}}_{attn}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{exp}}(t) - w_2 \iint g\left(u^{\text{atn}}(x,y)\right) \; dx \; dy \\ &+ w^{\text{pre}}_{attn} \; g\left(u^{\text{pre}}_{attn}(t)\right). \end{aligned} \tag{8.3}$$

The node's activation is pushed above threshold by input from the explore task of the scene representation (see Section 4.4). Once enough suprathreshold activation is present in the attention field, the inhibitory coupling (with weight w_2) pushes the precondition node below threshold again. The output of the precondition node, $g\left(u^{\text{pre}}_{\text{attn}}\right)$, provides an additional inhibitory input to the intention node of the *recognize* behavior (Equation 7.7).

Although the saliency mechanisms driving attention are only heuristics, they already provide coarse information on the location of attended objects. Ideally, these objects should be at the center of the attended locations. This is reflected by an additional broad Gaussian input that is added to the first layer shift estimation field. This input is relatively weak and provides bias toward the center of the image early in the recognition process. Stronger evidence that arises during the convergence process of the object recognition system may override this bias.

8.3 Interacting with the supervisor

The object recognition system reviewed in Chapter 3 can learn new objects quickly. However, the learning process in that implementation is controlled externally, either by a program or the user (Faubel and Schöner, 2009, 2010; Lomp et al., submitted 2016). Learning thus requires a fixed sequence of steps, wherein an image is first presented along with information on the pose and identity of the object, followed by a period in which the weights are adapted, in turn followed by a reset.

In this section, I describe an extension of the behavioral organization structure described in Chapter 7 and 8. This extension integrates the learn-

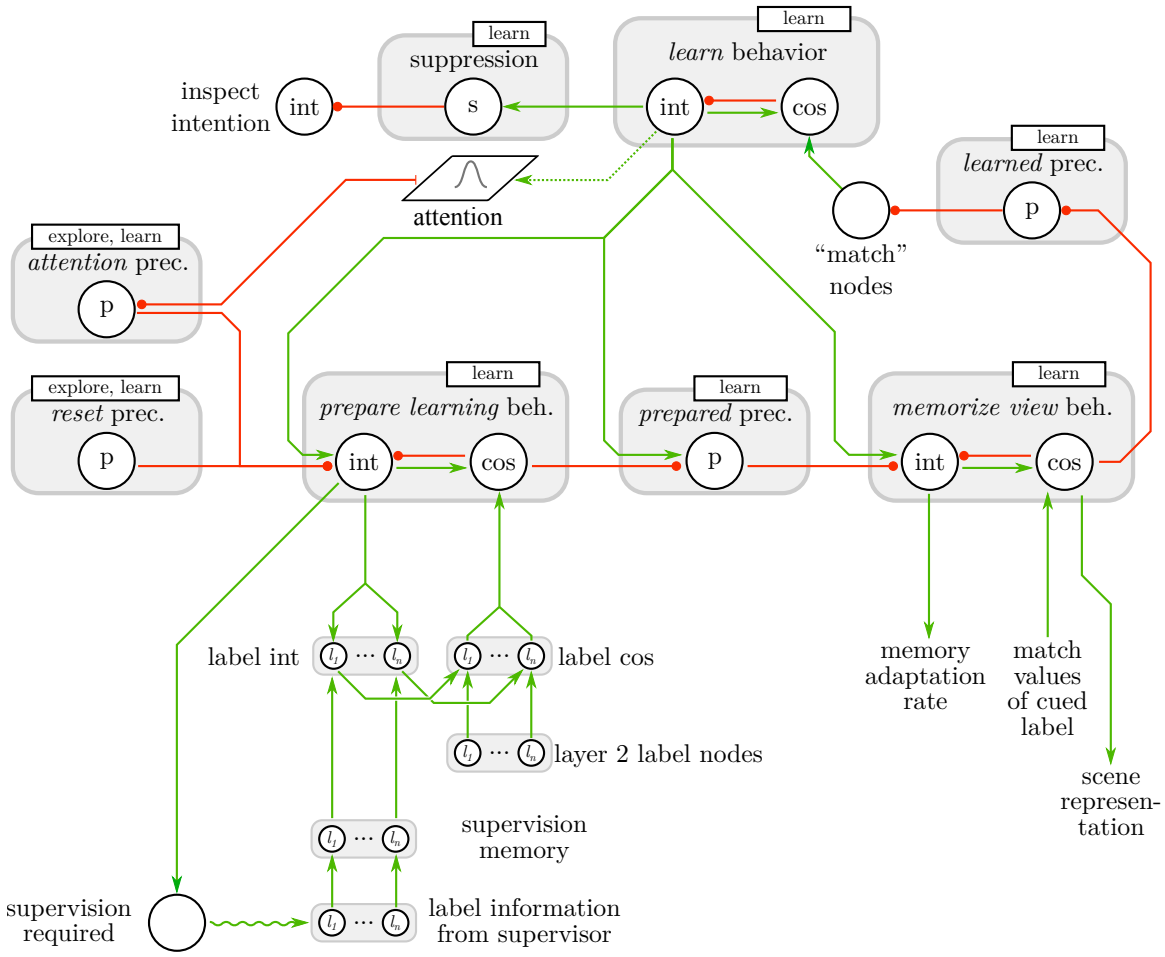


Figure 8.2: An overview of the behaviors involved in the learning process.

ing process into the neural-dynamics of the architecture. Users may activate a learning mode, in which the architecture sequentially scans objects in the scene, querying label information from the supervisor (that is, either the user or an algorithm) when a to-be-learned object is attended. When the supervisor provides this information, the learning process continues autonomously and repeats until all objects in the scene have been learned. The behavioral organization underlying this process is sketched in Figure 8.2 and described formally in the following subsections.

8.3.1 The *learn* behavior

The learning mode is enabled by a task *learn*. This task activates the precondition *location attended* (described in Section 8.2) by providing additional excitatory input to the node. It also activates a behavior *learn* (top of Figure 8.2). The intention node of this behavior follows the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{int}}_{\text{lrn}}(t) = &- u^{\text{int}}_{\text{lrn}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{lrn}}(t) - w_2 \; g\left(u^{\text{cos}}_{\text{lrn}}(t)\right) \\ &+ w^{\text{int}}_{\text{lrn}} \; g\left(u^{\text{int}}_{\text{lrn}}(t)\right), \end{aligned} \tag{8.4}$$

where $s^{\text{task}}_{\text{lrn}}(t)$ is the input from task learn. When the behavior's intention node becomes active, it excites a suppression node,

$$\begin{aligned}\tau \dot{u}^{\text{sup}}_{lsup}(t) = &- u^{\text{sup}}_{lsup}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{lrn}}(t) + w_2 \; g\left(u^{\text{sup}}_{\text{lrn}}(t)\right) \\ &+ w^{\text{sup}}_{lsup} \; g\left(u^{\text{sup}}_{lsup}(t)\right), \end{aligned} \tag{8.5}$$

which inhibits the *inspect object* intention node, $u^{\text{int}}_{\text{ins}}$, with an additional inhibitory input to Equation 4.18 (see page 51). Suprathreshold activity in the intention node also prompts the scene representation architecture to attend a previously unattended location by providing excitatory input to the attention field (indicated by the dotted arrow in Figure 8.2; see Section 8.3.4 for details). It also activates a chain of behaviors responsible for organizing the steps of the learning process (described in more detail below). The first step is a reset mediated by the *reset* behavior described in Section 7.1. It is followed by the *prepare learning* behavior, which queries label information from the supervisor. Finally, the *memorize view* behavior is responsible for learning the current object view. The sequence between these behaviors is brought about by a set of precondition nodes activated by the *learn* intention node.

The condition of satisfaction for the *learn* behavior follows the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{cos}}_{\text{lrn}}(t) = &- u^{\text{cos}}_{\text{lrn}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{lrn}}(t) + w_2 \; g\left(u^{\text{int}}_{\text{lrn}}(t)\right) + g\left(u^{\text{lma}}(t)\right) \\ &+ w^{\text{cos}}_{\text{lrn}} \; g\left(u^{\text{cos}}_{\text{lrn}}(t)\right), \end{aligned} \tag{8.6}$$

where u^{lma} is the activation of the label match node (analogous to the color match node described by Equation 4.20 on page 45).

8.3.2 The *prepare learning* behavior

To learn an object view, the supervisor must provide a label.[2] The *prepare learning* behavior (center of Figure 8.2) ensures that such information is present whenever a location is attended. The dynamics of the behavior's intention node reads

$$\begin{aligned}\tau \dot{u}^{\text{int}}_{\text{prep}}(t) = &- u^{\text{int}}_{\text{prep}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{lrn}}(t) + w_2 \; g\left(u^{\text{int}}_{\text{lrn}}(t)\right) \\ &- w_3 \; g\left(u^{\text{pre}}_{\text{attn}}(t)\right) - w_4 \; g\left(u^{\text{pre}}_{\text{res}}(t)\right) \\ &+ w^{\text{int}}_{\text{prep}} \; g\left(u^{\text{int}}_{\text{prep}}(t)\right). \end{aligned} \tag{8.7}$$

It is mainly driven by the input from the task and the *learn* intention, and is inhibited by the activation of the *location attended* precondition, $u^{\text{pre}}_{\text{attn}}$ (see Equation 8.3). Thus, it is deactivated whenever the peak in the attention field decays and nothing is attended. This happens, for example, when the currently attended object has been learned successfully, allowing the behavior to reactivate when multiple objects are present in the scene. Inhibition from the activation of the *reset* precondition node, $u^{\text{pre}}_{\text{res}}$ (see Equation 7.6), similarly ensures that when the input changes and a reset happens, the learning intention is reset as well.

The intention node excites a *supervision required* node that is meant as a shorthand for a system that communicates to the supervisor that label information for the attended location is required. The *supervision required* node has the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{sup}}(t) = &- u^{\text{sup}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; g\left(u^{\text{int}}_{\text{prep}}(t)\right) - w_2 \sum_l g\left(u^{\text{cosfl}}_l(t)\right) \\ &+ w^{\text{sup}} \; g\left(u^{\text{sup}}(t)\right), \end{aligned} \tag{8.8}$$

where $u^{\text{cosfl}}_l(t)$ is part of the condition of satisfaction structure of the behavior described below. When the *supervision required* node is active, the supervisor is expected to provide information on the label to be associated with the attended object via an input, $s^{\text{sup}}_l(t)$, to a set of label supervision memory

[2] Previously, pose information was also provided for evaluating the object recognition architecture. I forgo this here because it is only necessary to determine the precision of pose estimation, which I omit in the evaluation of the system. However, the same principles used for memorizing label information provided by the supervisor may be used to analogously represent pose information.

nodes. For a label l, the dynamics of the corresponding label supervision memory node is given by

$$\begin{aligned}\tau \dot{u}_l^{\text{supm}}(t) = &- u_l^{\text{supm}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s_l^{\text{sup}}(t) - w_2 \sum_{l' \neq l} g\left(u_{l'}^{\text{supm}}(t)\right) \\ &+ w^{\text{supm}} \; g\left(u_l^{\text{supm}}(t)\right). \end{aligned} \tag{8.9}$$

These nodes compete with each other (through inhibition of strength w_2), and at most one node may be active at any given time. The nodes are also in the working memory regime and thus memorize the information provided by the supervisor.

The goal, now, is to ensure that the label information provided by the supervisor is correctly represented in the architecture. Specifically, the nodes corresponding to the label cued by the supervisor must be active on the first and second layer nodes of the identity representation in the object recognition architecture. I realize this with a structure analogous to the intention and condition of satisfaction fields introduced in Section 2.3 (again, I substitute discrete nodes for the continuous feature space of the fields). The *label intention* nodes follow the dynamics

$$\begin{aligned}\tau \dot{u}_l^{\text{intf,l}}(t) = &- u_l^{\text{intf,l}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; g\left(u_l^{\text{supm}}(t)\right) + w_2 \; g\left(u_{\text{prep}}^{\text{int}}(t)\right) \\ &+ w^{\text{intf,l}} \; g\left(u_l^{\text{intf,l}}(t)\right), \end{aligned} \tag{8.10}$$

where, again, l is a label. The weights, w_1 and w_2, are chosen to only allow the node corresponding to the memorized information from the supervisor to become active if the *prepare learning* intention is active.

These nodes provide excitatory input to their corresponding label nodes in the first layer of label nodes in the object recognition system. Thus, the label cued by the supervisor is activated in the identity representation. The *label condition of satisfaction nodes* detect whether this has been achieved using the dynamics

$$\begin{aligned}\tau \dot{u}_l^{\text{cosf,l}}(t) = &- u_l^{\text{cosf,l}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; g\left(u_l^{\text{intf,l}}(t)\right) + w_2 \; g\left(u_{2,l}(t)\right) \\ &+ w^{\text{cosf,l}} \; g\left(u_l^{\text{cosf,l}}(t)\right), \end{aligned} \tag{8.11}$$

where $u_{2,l}(t)$ is the activation of node l on the second layer of the label nodes of the object recognition system. The weights, w_1 and w_2, are chosen to

only allow such a node to become active if both inputs are active, that is, if $g(u_l^{\text{intf,l}}(t)) \approx 1$ and $g\left(u_{2,l}(t)\right) \approx 1$.

Finally, the label condition of satisfaction nodes drive the condition of satisfaction node of the *prepare learning* behavior. The node's dynamics is

$$\begin{aligned}\tau \dot{u}^{\text{cos}}_{\text{prep}}(t) = &- u^{\text{cos}}_{\text{prep}}(t) + h + w_\xi\ \xi(t) \\ &+ w_1\ s^{\text{task}}_{\text{lrn}}(t) + w_2\ g\left(u^{\text{int}}_{\text{prep}}(t)\right) + w_3\ \sum_l g\left(u_l^{\text{cosf,l}}(t)\right) \\ &+ w^{\text{cos}}_{\text{prep}}\ g\left(u^{\text{cos}}_{\text{prep}}(t)\right). \end{aligned} \tag{8.12}$$

Here, the interaction through w_2 and w_3 only allows the condition of satisfaction to become active if the intention is active, and if there is an active *label condition of satisfaction* node.

8.3.3 The *learn view* behavior

A precondition for learning the current view is that information from the supervisor has been obtained, and that this information is represented in the object recognition architecture. This is expressed by a precondition node with activation, $u^{\text{pre}}_{\text{prep}}$ (prep for 'learning *prep*ared'), governed by the dynamics

$$\begin{aligned}\tau \dot{u}^{\text{pre}}_{\text{prep}}(t) = &- u^{\text{pre}}_{\text{prep}}(t) + h + w_\xi\ \xi(t) \\ &+ w_1\ s^{\text{task}}_{\text{lrn}}(t) + w_2\ g\left(u^{\text{int}}_{\text{lrn}}(t)\right) - w_3\ g\left(u^{\text{cos}}_{\text{prep}}(t)\right) \\ &+ w^{\text{pre}}_{\text{prep}}\ g\left(u^{\text{pre}}_{\text{prep}}(t)\right). \end{aligned} \tag{8.13}$$

The precondition node is activated as soon as the intention of the *learn* behavior becomes active, and is inhibited by the condition of satisfaction node of the *prepare learning* behavior. This precondition inhibits the intention node of the *learn view* behavior (center of Figure 8.2),

$$\begin{aligned}\tau \dot{u}^{\text{int}}_{\text{lrnv}}(t) = &- u^{\text{int}}_{\text{lrnv}}(t) + h + w_\xi\ \xi(t) \\ &+ w_1\ s^{\text{task}}_{\text{lrn}}(t) + w_2\ g\left(u^{\text{int}}_{\text{lrn}}(t)\right) \\ &- w_3\ g\left(u^{\text{cos}}_{\text{lrnv}}(t)\right) - w_4\ g\left(u^{\text{pre}}_{\text{prep}}(t)\right) \\ &+ w^{\text{int}}_{\text{lrnv}}\ g\left(u^{\text{int}}_{\text{lrnv}}(t)\right). \end{aligned} \tag{8.14}$$

The activation of this intention node controls the adaptation of the weights that store the current object view. That is, the control factor in Equation 3.38 (page 43) is set to $b_{\text{lrn}}(t) = H\left(u^{\text{int}}_{\text{lrnv}}(t)\right)$.[3]

[3]The Heaviside function is used here because the logistic function and its approximation approach zero but never actually reach it; this would mean that weights for all labels would

The condition of satisfaction node of the *learn view* behavior has the dynamics

$$\begin{aligned}\tau \dot{u}^{\cos}_{\text{lrnv}}(t) = &- u^{\cos}_{\text{lrnv}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{lrn}}(t) + w_2 \; g\left(u^{\text{int}}_{\text{lrnv}}(t)\right) + w_3 \; s^{\text{lrnv}}(t) \\ &+ w^{\cos}_{\text{lrnv}} \; g\left(u^{\cos}_{\text{lrnv}}(t)\right). \end{aligned} \tag{8.15}$$

When the behavior's intention node is active, the condition of satisfaction node may become activated by the input, $s^{\text{lrnv}}(t)$, which indicates sufficient similarity between the input pattern and the learned pattern for the cued label using the equation

$$s^{\text{lrnv}}(t) = \sum_F \sum_l H\left(g\left(u_{2,l}(t)\right) \; \text{match}^F_l(t) - \theta_F\right), \tag{8.16}$$

where $\text{match}^F_l(t)$ is defined by Equation 3.35 (on page 42), $u_{2,l}(t)$ is the activation of the view node for label l on the second layer of the object recognition system's label representation, and θ_F is a threshold parameter for feature channel F (color histograms, edge histograms on different channels, or shape; see Section 3.3). The condition of satisfaction is thus activated when the similarity between memory and input for the label provided by the supervisor exceeds a threshold for all feature channels. The condition of satisfaction node also excites both the spatial inhibition field (u^{spin} in Chapter 4) and the *label to working memory* nodes and *color to working memory* field (the latter is denoted by u^{c2wm} in Chapter 4, the former is the analogous structure for the label-feature channel).

The condition of satisfaction node inhibits a *learned* precondition node,

$$\begin{aligned}\tau \dot{u}^{\text{pre}}_{\text{lrnd}}(t) = &- u^{\text{pre}}_{\text{lrnd}}(t) + h + w_\xi \; \xi(t) \\ &+ w_1 \; s^{\text{task}}_{\text{lrn}}(t) - w_2 \; g\left(u^{\cos}_{\text{lrn}}(t)\right) \\ &+ w^{\text{pre}}_{\text{lrnd}} \; g\left(u^{\text{pre}}_{\text{lrnd}}(t)\right), \end{aligned} \tag{8.17}$$

which in turn inhibits the match nodes of the scene representation, preventing the scene representation from attending another object before learning of the currently attended object is complete.

8.3.4 Changes to the scene representation architecture

In its *explore* behavior, the scene representation sequentially scans salient locations in the input. It transitions from one object to the next when the

always adapt at least slightly, and the quality of the memory representation would thus decay over time.

feature values for the currently attended one are committed to memory. For learning, more explicit control of these processes is needed. That is, the learning behavior must be able to query the selection of a previously unattended object. This object must stay in the attentional focus until learning is complete.

This requires a modification of the scene representation architecture. The intention node of the *learn* behavior globally excites the attention field and thus induces the selection of a salient location. The looking working memory, as before, memorizes this attended location, preventing repeated inspection of the object.

The condition of satisfaction of the *memorize view* behavior feeds into the scene representation architecture. It excites the spatial inhibition fields of the color and label channel as well as the *color* and *label to working memory* fields. The currently attended object is therefore only memorized when it has been learned completely.

The *memorize view* behavior's condition of satisfaction also excites the condition of satisfaction of the *learn* behavior (see Equation 8.6), which inhibits the intention node of the *learn* behavior. When the latter becomes inactive, all nodes in the associated behaviors become inactive as well, which lets the condition of satisfaction node of the *learn* behavior fall below threshold. This, in turn, releases the inhibition of the intention node of the *learn* behavior. Because it still receives excitatory input from the task, it becomes active again, and a new object is attended and learned.

Chapter 9

Evaluation

In the present chapter, I describe the methods I used for quantitatively evaluating the performance of the integrated scene representation and object recognition architecture, as well as results of these evaluations.

9.1 Experimental protocol

Before evaluation, the architecture first learns to recognize the objects in the multi-object database (see Chapter 6) using either the single- or the multi-object training images. During the entire training procedure, the dynamics of the architecture are simulated without restarts. Images from one of the training sets of the multi-object database are presented to the architecture one by one. For each image, the training procedure begins by activating the task *learn*. When the *supervision required* signal is above threshold, the location of the maximum in the attention field is determined, and the label of the closest annotated object is provided by boosting the corresponding supervision node. After a fixed duration, the task *learn* is deactivated. An externally controlled homogeneous boost inhibits the looking working memory and space-feature working memory fields in the scene representation until their activation falls below threshold. After a fixed duration during which the preshape in the looking working memory is allowed to decay, training continues with the next image.

When the architecture is fully trained, it may be evaluated by presenting test images from the multi-object database. Again, the architecture's dynamics are simulated continuously for the entire duration of the test, and no hard resets take place. For the quantitative evaluation, images are presented to the system in an arbitrary order until each image is presented exactly nine times. For each image, the test begins by enabling the task *explore*.

Figure 9.1: Peaks at locations that are within the red circles are assigned to the object in the center of the circle. In case of ambiguities due to overlap, peaks are assigned to the object closest to their center.

The system may then explore the scene for a predetermined duration (to speed up the overall evaluation procedure, this duration is adapted to the number of objects in the image, allotting 20 seconds per object rather than using the same duration for all trials). When this duration is exceeded, the current contents of the space-label working memory are recorded for evaluation. The explore task is deactivated, and as in training, a homogeneous external boost inhibits the looking working memory and space-feature working memory fields in the scene representation. The process waits for a fixed duration to let the looking working memory preshape decay. Afterwards, testing continues with the next image.

9.2 Evaluation criteria

Only the space-label working memory is evaluated. Each recorded working memory state, obtained at the end time of the trial marked by t_{end}, is summed to obtain

$$P(x, y) = \sum_l g\left(u_l^{\text{slwm}}(x, y, t_{\text{end}})\right). \tag{9.1}$$

In addition, the label for each location is determined as the maximally active label at that location:

$$L(x, y) = \arg\max_l \left\{ g\left(u_l^{\text{slwm}}(x, y, t_{\text{end}})\right) \right\}. \tag{9.2}$$

The peaks in P are clustered to obtain peak locations, $\mathbf{x}_i = (x_i, y_i)$ (see Appendix B for details on the clustering approach). Together with the label information in L, this yields a list of working memory entries, $m_i = (\mathbf{x}_i, L(x_i, y_i))$.

Each working memory entry is matched to the annotated objects in the scene by determining the object closest to the entry's location. This yields the closest annotation, $m^* = (\mathbf{x}^*, l^*)$ that indicates an object with label, l^*, with its center at $\mathbf{x}^* = (x^*, y^*)$. If the distance between match and annotation is below a threshold, θ, that is, if $|\mathbf{x}_i - \mathbf{x}^*| > \theta$, the match is counted, otherwise, it is discarded from evaluation. For the experiments I report here, I choose $\theta = 150$px (Figure 9.1 illustrates the resulting circle for objects in an exemplary test image). This radius is chosen to include even the largest objects in the database (for example, the *cookies* in Figure 9.1) in this circle. Ambiguities that may result from overlaps of the individual circles are resolved by assigning peaks to the object closest to them.

These matches serve to calculate how well the objects in the input are represented by the system. For each object in the input, one of three cases may occur. In the first case, the object is not represented in working memory; that is, no working memory entry was found close enough to its location (I label this case *unrepresented*). This may be due to a failure of the scene representation to form a memory item, for example, because the object did not induce enough salience. Relatedly, this failure may also occur because the strongest saliency induced by the object does not coincide with the center of the object. As a result, the working memory peak may form too far away to be matched to the object. In the second case, exactly one working memory entry is matched to the object. The label of this entry may either match the annotated label (denoted *correct*) or not (denoted *incorrect*). In the third case, multiple working memory peaks are matched to one object. This case may be further subdivided into cases in which all working memory entries have the correct label (*overrepresented/correct*), cases in which some working memory items have the correct label (*overrepresented/mixed*), and cases in which no working memory item has the correct label (*overrepresented/incorrect*).

9.3 Results

In the single-object training case, the system formed a single correct peak for 67.1% of the objects in the database. Multiple correct peaks were associated with a single object in 7.7% of the cases. For 1.1% of the objects, both correct and incorrect peaks were associated with an object (in these cases, 54.8% of

the peaks had the correct label). Multiple incorrect peaks were associated with 0.2% of the objects, whereas a single incorrect peak was associated with 5.3% of the objects. No peaks were associated with 18.5% of the objects.

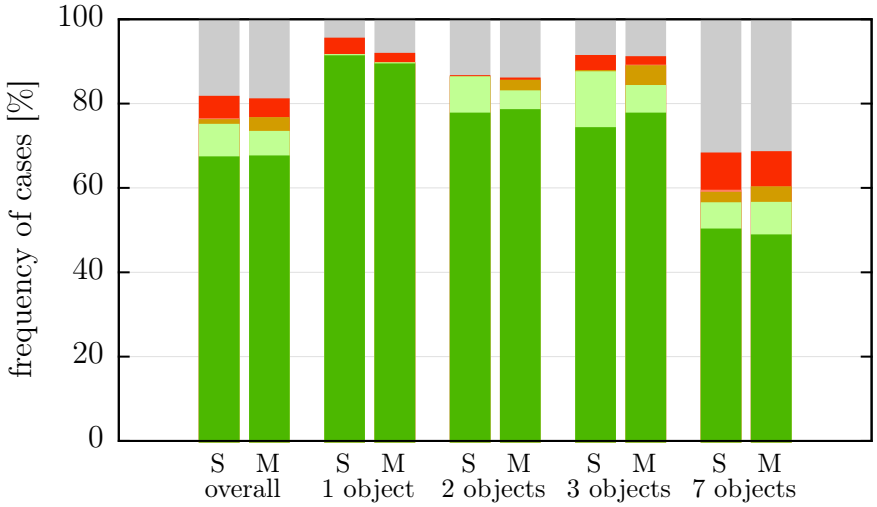
In the multi-object training case, the system formed a single correct peak for 67.4% of the objects. Multiple correct peaks were associated with a single object in 5.8% of the cases. Both correct and incorrect peaks were associated with 3.3% of the objects (in these cases, 51.6% of the peaks had the correct label). Multiple incorrect peaks were associated with none of the objects, whereas a single incorrect peak was associated with 4.4% of the objects. No peaks were associated with 19.1% of the objects.

Figure 9.2(a) shows the frequency of different outcomes over the number of objects in the image. The plot suggests that as the number of objects increases, performance deteriorates. I test the results for significance with Pearson's χ^2-test of independence, which confirms that the results are significant for the architecture that was trained with images containing a single object ($\chi^2(15, N = 2430) = 400.3$, $p < 0.001$; because the sample count is low for some combinations, I also performed Fisher's exact test, which yields a probability, $p \approx 0.0$, confirming the significance result of the χ^2-test). For the architecture trained using images of multiple objects, the results are also significant ($\chi^2(15, N = 2430) = 348.5$, $p < 0.001$).

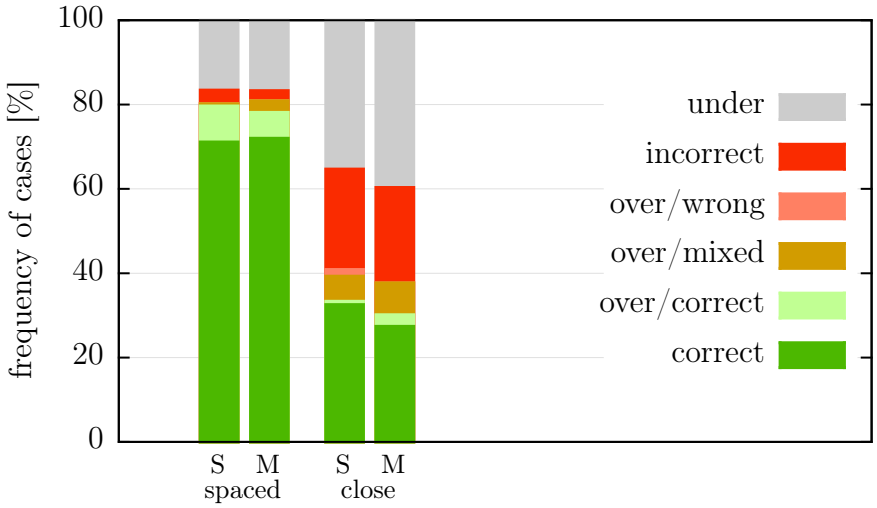
Figure 9.2(b) shows the frequency of different outcomes over the distance between objects. Here, the plot suggests that as the objects get closer together, performance deteriorates. To test whether there is a significant relation between the distance between objects and the outcomes, I again performed Pearson's χ^2-test of independence, which confirms that the relationship is significant for the architecture trained on single object images (again, I performed Fisher's exact test due to low sample counts and obtained $p \approx 0.0$) and also significant for the architecture trained using images containing multiple objects ($\chi^2(5, N = 2430) = 358.6$, $p < 0.001$).

Of the 2226 peaks generated by the architecture trained with images showing a single object, 17 could not be matched to an object by the criteria defined above. On average, these unmatched peaks were 189.4 px away from the nearest annotated object. The peaks that were matched to objects had, on average, a distance of 51.0 px. The average distance of matched peaks that encoded the correct label was 49.6 px, whereas the average distance of peaks encoding incorrect labels was 67.2 px.

The architecture trained with images containing multiple objects generated 2220 peaks over all test images, of which 18 could not be matched to an object. On average, the unmatched peaks had a distance of 285.1 px, whereas the matched peaks had an average distance of 51.6 px. The matched peaks that encoded the correct label had an average distance of 49.6 px, whereas



(a)



(b)

Figure 9.2: Frequency of different outcomes plotted (a) over the number of objects in the test image and (b) their spatial arrangement in the image. The left bars (marked "S") refer to the results from an architecture trained using images containing a single object, while the right bars (marked "M") refer to the results from an architecture trained using images containing multiple objects.

the peaks encoding an incorrect label had an average distance of 72.8 px

From the plots, the distributions of outcomes for the different training types appear similar. However, Pearson's χ^2-test of independence for the overall outcomes (the leftmost two bars of Figure 9.2(a)) reveals that, though the differences are small, the results are significant ($\chi^2(5, N = 4860) = 37.0$, $p < 0.001$).

Figure 9.3 further illustrates how well individual objects are recognized by the system. The *brush* posed the greatest difficulty and was rarely recognized and represented correctly. Likely, this is caused by its low saturation, leading to low saliency and a sparsely populated color histogram.

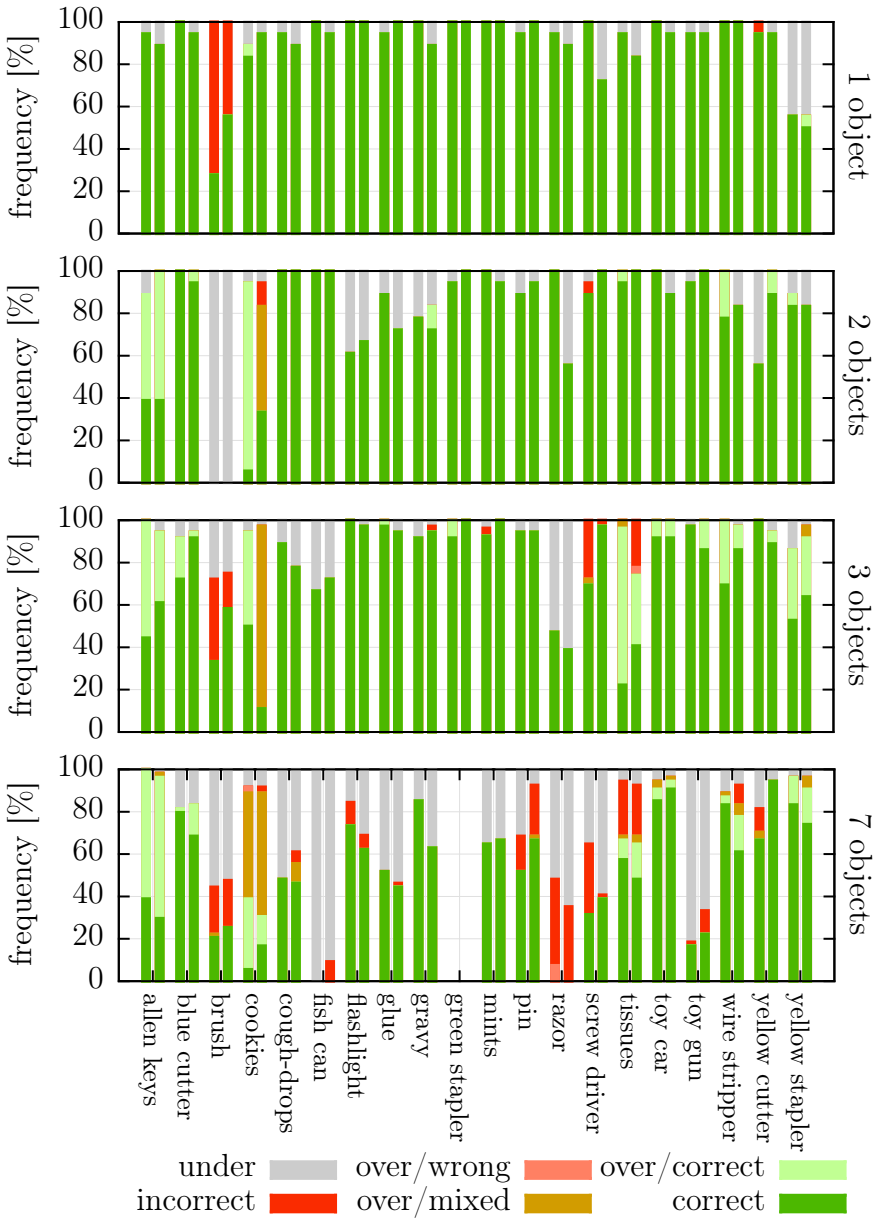


Figure 9.3: The frequency of different outcomes for each object, plotted separately for images containing one, two, three, or seven objects. As in previous plots, the left bars for each object refer to the results from an architecture trained using images containing a single object, while the right bars refer to the results from an architecture trained using images containing multiple objects. The bars for the green stapler are missing in the seven objects plot because this combination does not occur in the database.

Chapter 10

Discussion

In the present chapter, I first discuss exemplary cases in which the system made errors, followed by a discussion of the results and conclusions.

10.1 Examples of errors

Figure 10.1 shows four exemplary scenes onto which the working memory entries read out after scanning the scene have been superimposed. The figure demonstrates different possible outcomes of the scanning process: In Figure 10.1(a), all objects have been recognized correctly. However, two spatially distinct peaks have formed for the *Allen keys* (this is the case labeled *correct/over*). Figure 10.2 illustrates the cause for this duplicate entry. The elongated shape of the object induces above-average saliency over an equally elongated region (Figure 10.2(a)). When the object is attended, the attention field forms a peak close to the center of the object (Figure 10.2(c)), and this peak increases activation in the looking working memory that is too small to cover the entire salient region induced by the Allen keys (Figure 10.2(b)). Consequently, the attention field may still form further peaks for the object, which may therefore be inspected multiple times. Peaks from later inspections repel working memory peaks, leading them to drift and thus cause the locations of the working memory observed in the figure. All other objects in the image are sufficiently small, and therefore only a single peak forms that is close to their visual centers (labeled above as *correct*).

Figure 10.1(b) shows two different types of errors. The *brush* was incorrectly recognized as the *yellow stapler*, an object with a similar color distribution (this is labeled the *wrong* case above). The *pin* and *mints* were not attended (labeled *under* in the results section). This likely happened because the looking working memory preshape from previous trials did not

decay sufficiently to allow inspection of these objects. Scanning the scene again with a preshape that is initially zero everywhere results in the working memory entries shown in Figure 10.3. Although not all objects have been recognized correctly in this case, at least one peak formed for each object.

Figure 10.1(c) shows another type of error (labeled *mixed* above): the *cookies* induced working memory entries at two distinct locations, again due to their aspect ratio and size. One of these working memory entries has the correct label. However, the other has the wrong label, *screw driver*, presumably because both objects are predominantly yellow in color.

Finally, Figure 10.1(d) shows another issue that occurs when objects are located close together. In this case, multiple objects are included in the region used as input to the object recognition architecture. The figure illustrates this for the *pin* (labeled incorrectly as *yellow cutter*). The blue-white square indicates the input region of the object recognition system. The yellow cutter is still inside this region, and the recognition is therefore not necessarily incorrect, but rather refers to the wrong object.

One problem, exhibited also in Figure 10.1(b) and Figure 10.3, is that the original scene representation architecture models the process of scanning a single scene, but not the switch to another scene. Instead, the scene representation has an updating mechanism (change detection through the match and mismatch nodes in the three-layer structure described in Chapter 4). However, existing peaks and preshape inhibit attentional fixations on objects that occur at the same location previously occupied by other objects. For new objects in these locations to be properly attended, some time has to pass during which both the preshape and working memory peaks are allowed to decay. This increases the time required for a full evaluation of the architecture. To speed up processing, I addressed this with the external inhibitory boost that clears working memory described above, as well as a wait time between trials. However, the time allotted may not suffice for preshape to decay in all cases.

The slow decay of the preshape is a result of parameterization, and in previous scene representation implementations, preshape decays faster. In the implementation of the combined model, this decay time is extended to compensate for the longer time required to extract label information, because the faster decay would otherwise lead to a decay of the preshape before all objects in the scene are inspected, leading the system to only inspect a small number of objects before reinspecting the first.

10.2 Conclusion, related work and outlook

When individual objects are shown, the recognition rates of the combined architecture are comparable to the object recognition architecture on its own (see results in Faubel and Schöner, 2009, 2010; Lomp et al., submitted 2016). This suggests that neither the behavioral organization of the recognition processes nor the integration of the two architectures negatively impacted the functionality of the object recognition system. There is also a clear relationship between the number of objects and the performance. This can be explained because an increasing number of objects also implies an increased likelihood of distractors entering the input region of the object recognition architecture. Moreover, there are limits on the capacity of the working memory fields of the scene representation architecture (Zibner et al., 2010; much like there are limits to the working memory capacity of humans). Also, saliency may no longer be placed close enough to an object center when objects are close to each other. Further evidence for this is given by the large discrepancy of the recognition rates for the *spaced* and *close* category images.

Ideas similar to attention and saliency have previously been used in computer vision (a review can be found in Borji and Itti, 2013). The object detection system described by Viola and Jones (2001) uses a pipeline of filters to process the image using a sliding window approach. Filters at the start of the pipeline are good at detecting a large portion of faces, but at the same time falsely detect many non-faces. Along the pipeline, the rate of false detections decreases, and the final classification therefore achieves a high accuracy. This approach allows the system to discard non-target regions early in the processing hierarchy, reducing the number of filter applications down the pipeline. As a result, the full pipeline is only applied to a few subwindows of the image, saving computation time. Implicitly, the early stages of this pipeline are a form of saliency and attentional selection.

Miau et al. (2001) incorporate an explicit saliency computation that uses biologically inspired features, a winner-take-all network and an inhibition of return mechanism to determine a *scan path*, which determines a sequence of image regions that is passed to an object detection system. They test two different systems for object detection, one based on support vector machines, and one based on the HMAX model. They show that attention greatly reduces the amount of data that must be processed by the system with only a small loss of precision, and show that good performance in pedestrian detection may be obtained on real-world images with support vector machines, whereas the HMAX model is tested with artificial stimuli only. This approach again supports the view that attention reduces the computational demand for the vision system.

Walther and Koch (2006) present a similar combination of saliency-guided attention with the HMAX model, aiming to provide a more biologically plausible approach. Saliency is again extracted using basic image features and a winner-take-all network and an inhibition of return mechanism. To perform recognition, the image is first scanned by the saliency network. The resulting salient locations are recorded and presented sequentially to a feedforward HMAX system, which detects paperclips in one experiment and faces in another with high accuracy. Performance depends on the distance between the objects; if the objects are close together, the system also experiences problems properly recognizing the target objects, much like the issues encountered for the combined system I presented above.

Later, Walther and Koch (2007) lifted this approach to a more general framework for object recognition with attention. This framework describes such a combined architecture as the extraction of different feature maps from the input that are further processed to form the basis for object recognition. An attentional map modulates which part of the input contributes to feature extraction and may also incorporate top-down feedback based on, for example, the representation of a target object.

The literature cited thus far shows that saliency can reduce the computational demands of object recognition without negatively impacting performance. By contrast, Han and Vasconcelos (2010) investigate the impact of saliency on recognition performance. Similar to the previously cited literature, they also build on the HMAX model, but replace the model's first layer, the simple cells, by different saliency networks based on statistical inference. They compare performance of the network with different saliency mechanisms and find that the use of a saliency mechanism may enhance recognition performance for a real-world dataset.

All of the reviewed approaches have shown benefits of incorporating saliency and object recognition. However, a biologically plausible mechanism for representing memory of the recognitions such as the label-feature fields in the model I present here is absent from all of these models. Similarly, though the networks model the process of sequential scanning, object recognition is usually not integrated into this process.

I have demonstrated a system that successfully integrates object recognition, memory and attention into a single process. This system is capable of learning objects presented in a scene, and it recognizes these objects in test scenes, inspecting each object and associating its location with the recognized label in a working memory field.

Even though the learning of objects is not fully realized without input from the user, all that is required is the input of a label for the currently attended object. If this label is thought of as a name for the object, this is

not an unreasonable expectation; in a robotic scenario, this could mean that the robot points to an object, indicating the need for a label, which the user provides verbally.

Another remaining issue is that the combined system needs to be explicitly switched into a training mode. This could be replaced by a novelty detection approach such as that of adaptive resonance theory (Carpenter and Grossberg, 2003), though the details of such a mechanism are left for future work.

Otherwise, training seems to work well even when multiple objects are in the image, meaning that at least an explicit training by placing individual objects at a fixed position, orientation and scale in the camera image is no longer necessary. Compared to the manual approach in Faubel and Schöner (2009, 2010) and Lomp et al. (submitted 2016), this greatly reduces the effort required for training the architecture.

The tests of the architecture have thus far only been performed with static images. A more rigorous test of both the learning and recognition capacities in an actual robotic scenario in which the architecture, through the robot, interacts with a human user is desirable. A potential scenario could be solving a common task, such as the construction task presented by Bicho et al. (2009). In their approach, the authors kept the vision architecture simple, but they already use an object memory layer to keep track of the position of relevant objects, much like the space-feature fields used in the present work.

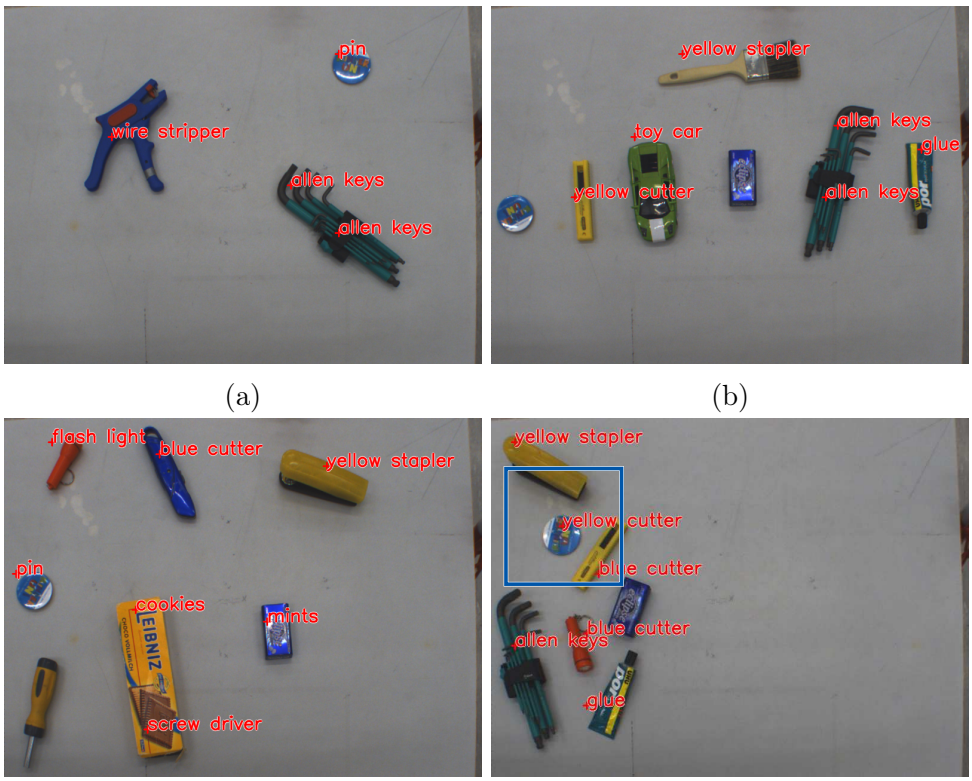


Figure 10.1: Visualizations of the working memory contents after scanning the shown scenes for a fixed duration. Red crosses indicate the location of the working memory entry (interpolated from the lower-resolution working memory field). Recognized labels are indicated by the text next to the crosses. The blue square indicates the input region of the object recognition system for the example of the working memory entry at the square's center.

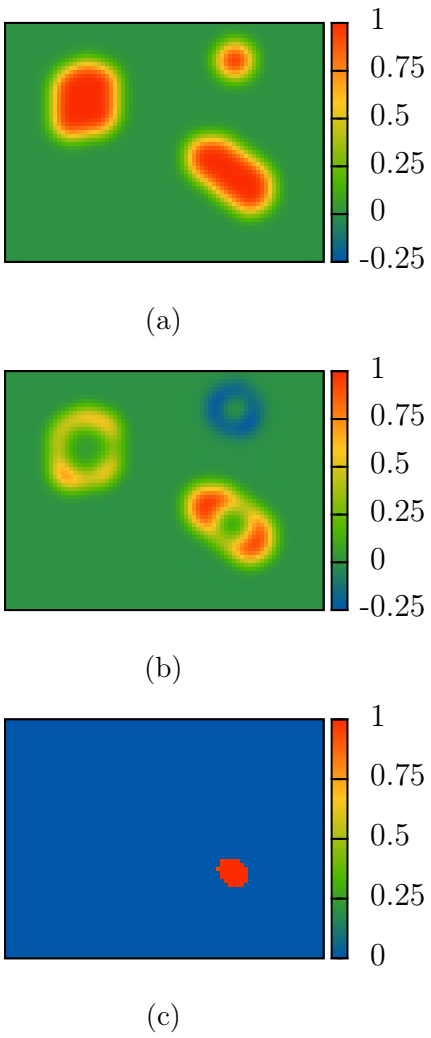


Figure 10.2: Excerpt of the architecture's state during the exploration of the scene shown in Figure 10.1(a). (a) shows the color saliency map prior to the inspection of any object. (b) shows the saliency combined with inhibition from the looking working memory after all objects have been inspected. (c) shows the output of the attention field during the first fixation on the Allen keys.

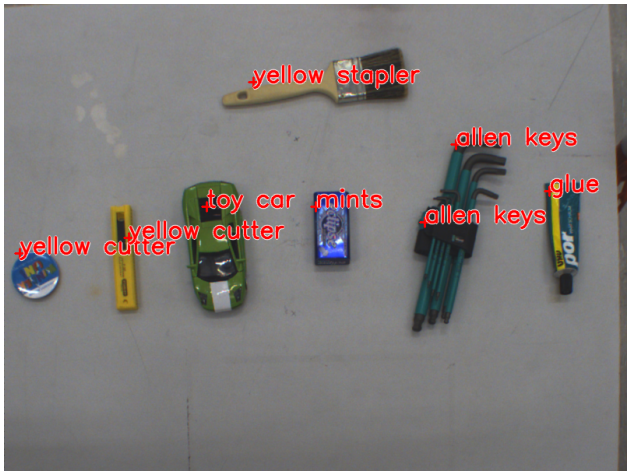


Figure 10.3: Result of scanning the scene from Figure 10.1(b) with initially flat preshape for the looking working memory.

Part III

Object recognition based on space-feature patterns

Introduction

My goal in Part III is to investigate the biological plausibility of the histogram based object recognition system presented in Chapter 3. In this system, pose is represented by neural fields that inherit biological plausibility by design. However, the biological plausibility of the pose transformations is not as obvious because they require multiplying the activation of the pose fields with the input pattern, which deviates from common neural mechanisms in dynamic field theory. However, such multiplicative connectivity has been used in other neural network models before (Olshausen et al., 1993, for example, describes a system in which attention provides a multiplicative gain value that routes information at an attended location to the recognition system).

An alternative transformation approach is described by Schneegans and Schöner (2012). In this approach, two patterns provide input to a two-dimensional transformation field. The first pattern is the pattern to be transformed. The second pattern is the transformation parameter and is analogous to the fields representing pose in the object recognition system described in Chapter 3. Both patterns provide a ridge input along one of the dimensions of the transformation field. Peaks form where these inputs overlap, and a diagonal readout yields the input pattern shifted by the transformation parameter. For this approach, the input pattern and transformation parameter must be encoded in fields, or at least having an analogous structure. This is not the case for the localized histograms, because they encode the rate of occurrence of individual feature values, rather than encoding presence or absence of individual values.

One of my goals in this part is therefore to provide a transformation architecture that uses patterns more closely related to the activation and output patterns of dynamic neural fields. I refer to these patterns as space-feature patterns because—analogous to neural fields—they are activation patterns defined over two spatial dimensions and a feature dimension. Such patterns may be transformed using the principles described in Chapter 3, using an approach that combines concepts from both the shape-based channel as well as the histogram based channels. In Chapter 11, I describe this combined ap-

proach, which is based on a prototypical implementation described in Lomp et al., 2014.

Space-feature patterns do not only serve to increase the biological plausibility of the transformation process, but also of the object representation, because the localized feature histograms described in Chapter 3 are largely invariant under changes in the spatial arrangement of the constituent feature values because they aggregate feature information over space. Thus, a set of feature values may yield the same localized histogram in different spatial arrangements, whereas, by contrast, there is only a small degree of tolerance to changes in the spatial arrangement of feature values in the brain (Vogels, 1999).

The main discriminative power of the histogram based recognition system comes from color. Other feature channels mainly facilitate pose estimation (see Lomp et al., submitted 2016 and the results in Chapter 12). How does this compare to object recognition in the brain? Certainly, color supports object recognition. For example, color has been shown to improve performance in search tasks and conditions in which shape was degraded (Markoff, 1972), as well as in naming tasks (Ostergaard and Davidoff, 1985). However, Biederman and Ju (1988) have demonstrated a paradigm in which color does not play a role for object recognition unless shape information is unreliable or degraded (but see Wurm et al., 1993 for criticism), and many of the experiments in the Biederman line use simple line drawings which are recognized successfully by participants (for example, Biederman, 1987; Biederman and Cooper, 1992). Other experiments have shown the ability to recognize novel stimuli in the absence of color information (see, for example, Nazir and O'Regan, 1990), and our ability to recognize grayscale images also suggests that we can recognize objects without color information. On the anatomical side, the prevalence of simple and complex cells found in the visual cortex that primarily react to lines, edges and similar features (Hubel and Wiesel, 1962, 1968; Wandell, 1995) also speaks to the importance of representations not based on color alone. To date, the exact role of color for object recognition remains unclear (Hagen et al., 2016), but it does not seem to be as critical for discrimination as it is in the histogram-based object recognition approach.

Another issue is that during the initial stages of matching color histograms, the histograms are effectively extracted from the whole image due to the unspecific position estimate. These histograms are highly distinctive for each object, and many objects can therefore already be discarded from the match even before the pose is estimated. Empirically, I have found that the initial guess of the system often already favors the correct recognition, which greatly simplifies pose estimation. This can also be seen in Lomp et al., sub-

mitted 2016, where pose estimation performance has been shown to improve only marginally when the correct label is provided for the entire recognition process. In addition, the localized color histograms are only affected by shifts, but are invariant to rotations. Taken together, this means that the interplay between pose and identity estimation has not yet been fully tested.

The first kind of space-feature patterns I describe are therefore based on edge orientations rather than color features. In Chapter 12, I describe how such patterns may be extracted, and how this representation impacts the structure of the transformation and matching architecture.

In Chapter 13, I extend the edge orientation based architecture by an additional space-color pattern channel. In contrast to the localized histogram architecture, space-color patterns are a more holistic image representation, so that the object's full shape (in the two-dimensional image plane) may be extracted. I show that this can be used to determine which parts of a matched input image belong to the matched object.

Finally, Chapter 14 concludes the present part with a discussion of the approach and the results obtained for the different types of space-feature patterns.

Chapter 11

Object recognition based on space-feature patterns

In the present chapter I first introduce how space-feature patterns may be transformed and matched in a framework analogous to the one presented in Chapter 3. Concrete formalizations for both the extraction of space-feature patterns and any modifications to the transformation and matching architecture that may ensue are described in the two chapters following the present one.

The neural dynamics of pose representation in the systems presented here are mostly analogous to those presented in Chapter 3, with the addition of the behavioral organization components described in Chapter 7, as well as a new competition scheme described in Section 11.2. In addition, two new neural fields are added for representing the estimated scale of the object. Their dynamics are analogous to the position and orientation fields, merely the dimension over which they are defined is changed. The scene representation architecture is not included here, as the focus in the present part is to investigate the capacity to recognize individual objects based on space-feature patterns.

11.1 Pose-transformations and matching for space-feature patterns

A space-feature pattern is a function $P_F(x, y, f, t)$ with $P_F : \mathbb{R}^3 \times \mathbb{R}^+ \rightarrow \mathbb{R}$, where F indicates a feature channel such as color, x, y are image coordinates, f is a feature value, and $t \in \mathbb{R}^+$ represents time. The pose-transformation and matching approach for space-feature patterns is again divided into two separate paths, the *bottom-up* path (indicated in equations by a subscript

bu) and the *top-down* path (indicated in equations by a subscript td), which are analogous to the two paths in the pose-transformation and matching architecture described in Chapter 3.

11.1.1 Pose-transformations in the bottom-up path

In analogy to the bottom-up path described in Chapter 3, the goal in this section is to align the input space-feature pattern with the space-feature patterns in memory. The space-feature pattern is therefore first transformed by the current shift estimate, p_{sh}, which assigns a weight between zero and one to each possible shift (see Section 3.2.1). The transformation is realized by a convolution that is only applied to the spatial dimensions, x and y:

$$P^{\text{sh}}_{\text{bu},F}(x, y, f, t) = \iint p_{\text{sh}}(x - x', y - y', t)\ P_F(x', y', f, t)\ dx'\ dy'. \tag{11.1}$$

When the recognition process is converged so that only a small region of the shift estimate contains nonzero values, this convolution shifts the input pattern so that the object is at the same position as in the best matching learned view.

For applying rotation, the pattern is transformed to log-polar coordinates. Again, this transformation is only applied to the spatial coordinates x and y. Thus, the result of the shift transformation in log-polar coordinates, where ϕ is the angle and ρ the distance from the origin, is given by

$$P^{\text{sh}}_{\text{bu},F}(\rho, \phi, f, t) = P^{\text{sh}}_{\text{bu},F}\big(\exp(\rho)\sin(\phi),\ \exp(\rho)\cos(\phi),\ f, t\big). \tag{11.2}$$

This pattern is transformed by the current rotation estimate, p_{rot}, which assigns a weight between zero and one to each possible rotation (see Section 3.2.1), by a convolution that is only applied to the angle dimension of the log-polar representation:

$$P^{\text{rot}}_{\text{bu},F}(\rho, \phi, f, t) = \int p_{\text{rot}}(\phi - \phi', t) P^{\text{sh}}_{\text{bu},F}(\rho, \phi', f, t)\ d\phi'. \tag{11.3}$$

When the recognition process is converged, this equation shifts the pattern along the angle dimension, thus rotating the input pattern in Cartesian space so that the object has the same orientation as the object in the best matching learned view. If the feature, f, is not invariant under rotation, further transformations may need to be applied (see Chapter 12).

Because ρ represents the logarithmic distance from the center of the coordinate system, scaling by a factor may also be realized as a shift. The rotated pattern is thus convolved along the ρ-dimension by the current scale

estimate, p_{sc}, which again assigns a weight between zero and one to each scale (see Section 3.2.1). The scaled pattern is thus

$$P^{\text{sc}}_{\text{bu},F}(\rho,\phi,f,t) = \int p_{\text{sc}}(\rho-\rho',t)\; P^{\text{sh}}_{\text{bu},F}(\rho',\phi,f,t)\; d\rho'. \tag{11.4}$$

When the recognition process is converged, this equation scales the input pattern in Cartesian space so that the object has the same scale as the object in the best matching learned view.

This pattern in Cartesian coordinates,

$$P^{\text{sc}}_{\text{bu},F}(x,y,f,t) = P^{\text{sc}}_{\text{bu},F}\left(\log\left(\sqrt{x^2+y^2}\right),\; \arctan(x,y),\; f,t\right), \tag{11.5}$$

is the input pattern, transformed by the current shift, rotation and scale estimates. Although this pattern can be matched against the memorized space-feature patterns, $P^{\text{mem}}_{F,l}(x,y,f,t)$ using the correlative matching function described in Chapter 3, I use a modified matching approach which I describe next.

11.1.2 Matching space-feature patterns

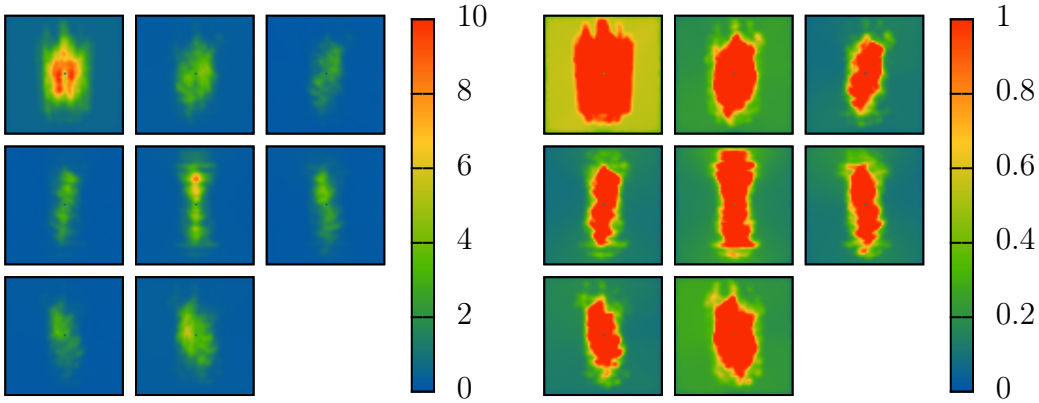
With the correlative matching approach used for localized feature histograms (see Chapter 3), the entire pattern contributes to the match. For the histograms, this is useful because the pattern does not distinguish between foreground and background. However, space-feature patterns may express the absence of feature information by locations which have activation values close to zero for all possible feature values. I define these locations as *background locations*. Matching should then only take into account locations at which either the top-down pattern indicates foreground, or at which the input pattern indicates the presence of (valid) feature information.

The matching of two space-feature patterns P_1 and P_2, therefore uses a mask,

$$M(x,y,t) = H\left(\sum_{i=1}^{2} H\left(\int P_i(x,y,f,t) - \vartheta\; df\right)\right). \tag{11.6}$$

This mask assigns the value zero to locations at which none of the patterns has valid feature information, and assigns the value one to locations at which either (or both) of the patterns represent valid feature values (as determined by a threshold, ϑ). Using this mask, the normalized pattern is given by

$$\hat{P}_i(x,y,f,t) = \begin{cases} \frac{1}{n_i(t)}\left(P_i(x,y,f,t) - m_i(t)\right): & M(x,y,t) > 0 \\ 0: & \text{otherwise,} \end{cases}, \tag{11.7}$$



(a) unsaturated superposition (b) saturated superposition

Figure 11.1: Superposition of space-edge patterns (described in detail in Chapter 12) of all learned views for the tabletop dataset (a) before and (b) after saturation. In the figures, each squares shows a slice of the space-edge map, each of which stands for a specific edge orientation (from left to right, top to bottom, 0° to 180°).

---

where $i \in \{1, 2\}$, the modified mean, m_i, is given by

$$m_i(t) = \frac{\iiint M(x, y, t)\ P_i(x, y, f, t)\ dx\ dy\ df}{\iint M(x', y', t)\ dx'\ dy'}, \tag{11.8}$$

and the modified norm, n_i, is given by

$$n_i(t) = \sqrt{\iiint \left(M(x, y, t)\ P_i(x, y, f, t)\right)^2\ dx\ dy\ df}. \tag{11.9}$$

Point-wise multiplication of two normalized patterns yields a match value which provides input to the label nodes described in Section 3.2.2. Analogously, cross-correlation of two patterns yields match values which provide input to the fields representing pose as described in Section 11.1.4.

11.1.3 Saturating the superposition of learned views

As in Chapter 3, the memorized patterns are superposed before pose transformations are applied in the top-down path. During preliminary experiments, I observed that in the initial phase of recognition, the different views would

often combine in such a way that a single, strong feature value dominated the entire superposition. Figure 11.1(a) illustrates this for a representation based on edge orientations. As a result of such dominant regions, the matching process often failed to form an initial position estimate that is sufficiently broad to allow for proper localization. To avoid such dominant regions, I introduce a saturation term for the superposition of the learned object views, which is calculated as

$$P_{\text{td},F}(x, y, f, t) = \min\left\{1,\ \sum_l p_l(t)\ P^{\text{mem}}_{F,l}(x, y, f, t)\right\}, \tag{11.10}$$

where $P^{\text{mem}}_{F,l}$ is the learned space-feature pattern for the object view indexed by the label, l. Figure 11.1(b) illustrates how this saturation affects the superposition of the memories.

11.1.4 Pose-transformations in the top-down path

In analogy to the architecture described in Chapter 3, the goal of the top-down path is to align the top-down pattern with the input and to determine match values for the pose parameters. The saturated memory superposition is therefore first transformed to log-polar coordinates and matched with the fully transformed pattern from the bottom-up path to obtain a real-valued match for scale,

$$\text{match}^F_{\text{sc}}(\rho, t) = \iiint \hat{P}_{\text{td},F}(\rho + \rho', \phi', f', t)\ \hat{P}^{\text{rot}}_{\text{bu},F}(\rho', \phi', f', t)\ d\rho'\ d\phi'\ df', \tag{11.11}$$

where the "hat" indicates normalization as described in Section 11.1.2. As before, this match value servers as input to the neural fields that represent the scale estimate (see Section 3.2.1).

To match the object's rotation, the superposition of the memorized space-feature patterns is scaled according to

$$P^{\text{sc}}_{\text{td},F}(\rho, \phi, f, t) = \int p^{\text{inv}}_{\text{sc}}(\rho - \rho', t)\ P_{\text{td},F}(\rho', \phi, f, t)\ d\rho', \tag{11.12}$$

where $p^{\text{inv}}_{\text{sc}}(\cdot)$ is the inverse of the scale estimate (see Section 3.1.2, in the description of Equation 3.7). The rescaled superposition is matched with the shifted input pattern,

$$\text{match}^F_{\text{rot}}(\phi, t) = \iiint \hat{P}^{\text{sc}}_{\text{td},F}(\rho', \phi + \phi', f', t)\ \hat{P}^{\text{sh}}_{\text{bu},F}(\rho', \phi', f', t)\ d\rho'\ d\phi'\ df', \tag{11.13}$$

where the "hat" indicates normalization as described in Section 11.1.2. This match provides input to the fields that represent rotation (see Section 3.2.1).

To match the shift between the memory representation and the input, the scaled superposition is rotated according to

$$P^{\text{rot}}_{\text{td},F}(\rho,\phi,f,t) = \int p^{\text{inv}}_{\text{rot}}(\phi-\phi')\; P^{\text{sc}}_{\text{td},F}(\rho,\phi',f,t)\; d\phi', \tag{11.14}$$

where $p^{\text{inv}}_{\text{rot}}(\cdot)$ is the inverse of the rotation estimate (see Chapter 3). Transformed back to Cartesian coordinates, the rotated superposition can be matched to the input with

$$\text{match}^{F}_{\text{sh}}(x,y,t) = \iiint \hat{P}^{\text{rot}}_{\text{td},F}(x+x',y+y',f',t)\; \hat{P}_{F}(x',y',f',t)\; dx'\; dy'\; df', \tag{11.15}$$

where the "hat" indicates normalization as described in Section 11.1.2. This match value is input to the fields that represent the shift estimate (see Section 3.2.1).

In the approach described in Chapter 13, I also make use of the inversely shifted top-down pattern given by

$$P^{\text{sh}}_{\text{td},F}(x,y,f,t) = \iint p^{\text{inv}}_{\text{sh}}(x-x',y-y',t)\; P^{\text{rot}}_{\text{td},F}(x',y',f,t)\; dx'\; dy', \tag{11.16}$$

where $p^{\text{inv}}_{\text{sh}}(\cdot)$ is the inverse of the shift estimate (see Chapter 3). When the label and pose estimates are converged, $P^{\text{sh}}_{\text{td},F}$ is the top-down pattern, aligned with the input pattern.

11.2 Gradually increasing competition in pose and identity representation

In the histogram-based architecture, ambiguous pose and label estimates only seldom arise because the color histograms are usually highly specific to individual objects (as discussed in the introduction to the present part). By contrast, edge orientation histograms, on their own, are far less discriminative (see Lomp et al., submitted 2016 and the results in Table 12.1) and are therefore more susceptible to ambiguities in the stimulus. With space-feature patterns, the additional information provided by the spatial arrangement, in part, resolves such ambiguities. However, during the initial stages of matching, this additional information is lost because the input pattern is

transformed by unspecific pose estimates which effectively removes the spatial arrangement. Pose estimates are therefore more prone to ambiguity at this early stage, and the system may converge towards a wrong solution. Although these solutions are not necessarily globally optimal in the sense of maximal matches values, the competitive dynamics may prevent the system from escaping from these states because the correct solution is no longer considered a viable candidate. This situation is, to an extent, analogous to the problem of local minima and maxima in optimization approaches.

In practice, this means that the system often converges to an incorrect estimate because a decision for that estimate is made too early. That is, alternative pose candidates are inhibited too strongly too early because an incorrect recognition temporarily yields higher match values with a transient pose estimate. A solution for a similar problem in an architecture that also uses a recurrent transformation and matching process has been discussed in Lücke et al. (2008) and Wolfrum et al. (2008). I adopt a similar solution by introducing an *inhibitory node* for the first layers of the label and pose representations. These inhibitory nodes gradually ramp up the amount of competition in the first layers of the pose and label representations and replace the global inhibition term in their dynamics. In the initial stage of recognition, competition is thus weaker, and ambiguous candidates are viable for a longer portion of the convergence process. This allows them to gain higher match values as parts of the pose estimates sharpen and potentially resolve ambiguities. The dynamics of the first layer of label nodes (Equation 3.16 on page 34) thus becomes

$$\begin{aligned}\tau_1 \dot{u}_{1,l}(t) = &- u_{1,l}(t) + h_1 + w_\xi \, \xi(t) \\ &+ s_{1,l}(t) - w^{\mathrm{inh,L}} u^{\mathrm{L,inh}}(t) \\ &+ w^{\mathrm{u_1}} \, g\left(u_{1,l}(t)\right), \end{aligned} \tag{11.17}$$

where $w^{\mathrm{inh,L}}$ is the strength of the coupling from the inhibitory node. The activation of the inhibitory node for the first layer of labels is denoted by $u^{\mathrm{L,inh}}(t)$. It is governed by the dynamics

$$\begin{aligned}\dot{u}^{\mathrm{L,inh}}(t) = &\quad \frac{g\left(s^{\mathrm{L,inh}}(t) - \theta\right)}{\tau_\mathrm{b}} \Big(- u^{\mathrm{L,inh}}(t) + s^{\mathrm{L,inh}}(t) \Big) \\ &+ \frac{1 - g\left(s^{\mathrm{L,inh}}(t) - \theta\right)}{\tau_\mathrm{d}} \Big(- u^{\mathrm{L,inh}}(t) \Big), \end{aligned} \tag{11.18}$$

with a threshold θ and input

$$s^{\mathrm{L,inh}}(t) = \sum_l g\left(u_{1,l}(t)\right). \tag{11.19}$$

The dynamics of the inhibitory node is a zero-dimensional analog for the preshape dynamics described by Equation 4.8 (page 48). When suprathreshold activation is present in the layer one label nodes, $g\left(s^{\mathrm{L,inh}}(t) - \theta\right) \approx 1$ and the activation relaxes to the input with the buildup timescale τ_{b}. When all labels are below threshold, $g\left(s^{\mathrm{L,inh}}(t) - \theta\right) \approx 0$ and the activation of the inhibitory node returns to zero on the fast decay timescale, $\tau_{\mathrm{d}} < \tau_{\mathrm{b}}$. This fast decay allows inhibition to quickly reset between two recognitions.

Analogous inhibitory nodes are added for the pose parameters. The dynamics of the first layers of the pose representation (see also Chapter 3) thus become

$$\begin{aligned}\tau_1 \dot{u}_1(r,t) = &- u_1(r,t) + h_1 + w_\xi\, \xi(r,t) \\ &+ s_1(r,t) - w^{T,\mathrm{inh}} u_T^{\mathrm{inh}}(t) \\ &+ \left[k^{\mathrm{u_1}} * (g \circ u_1)\right](r,t)\,,\end{aligned} \tag{11.20}$$

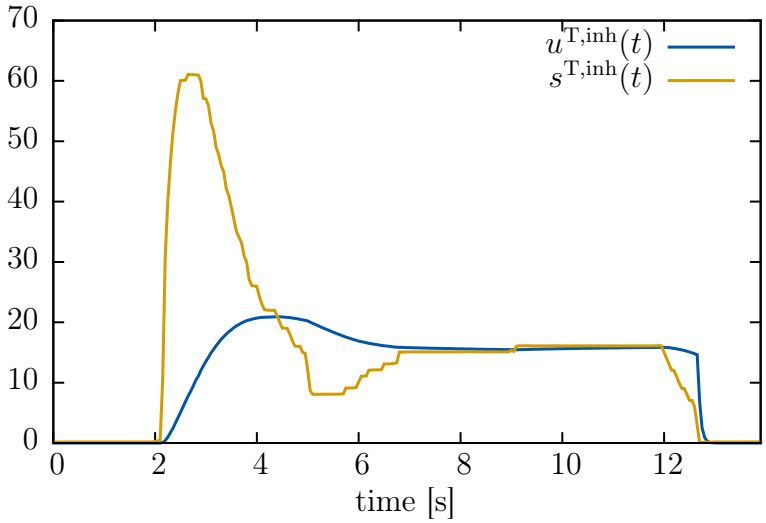
where T indexes one of the pose parameters (position, orientation and scale) and $w^{T,\mathrm{inh}}$ is the strength of the coupling from the inhibitory node to the field. The activations of the inhibitory nodes for the pose fields are denoted by $u_T^{\mathrm{inh}}(t)$. Their dynamics is analogous to Equation 11.19 with input

$$s^{\mathrm{T,inh}}(t) = \int g\left(u_1(r')\right)\, dr'. \tag{11.21}$$

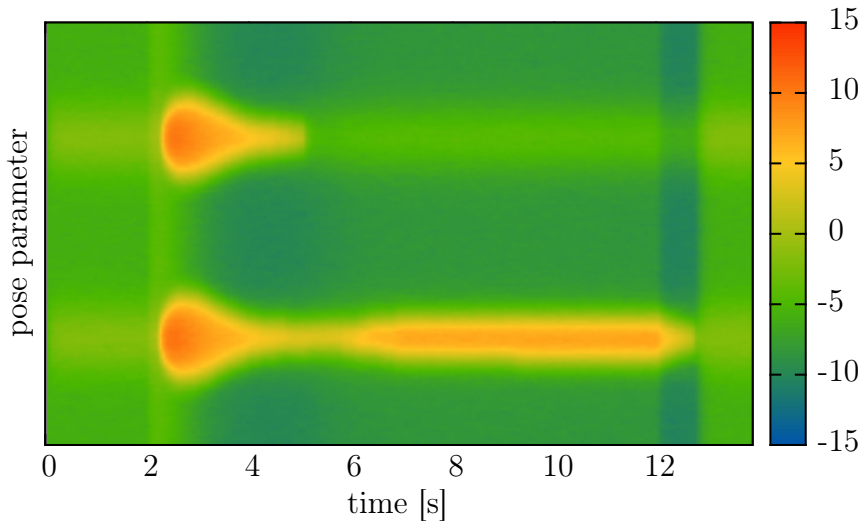
Figure 11.2 shows a numerical simulation of selection in an exemplary one-dimensional field driven by such an inhibitory node. The input of the field consists of two subthreshold Gaussians of equal amplitude and width, but at different locations. A homogeneous boost at $t = 2$ s drives the activation of the field above threshold. For the next three to four seconds, activation at both locations is above threshold and causes the inhibitory node's activation to rise slowly. This, in turn, creates increased inhibition which suppresses the entire field. At a critical point, the activation at one of the input locations falls below threshold. Because this reduces the overall activation of the inhibitory node, the active peak strengthens over time. Towards 13 seconds, the homogeneous boost is deactivated. This allows activation in the entire field to fall below threshold. The decay term of the inhibitory node takes effect, and the node's activation falls back to zero, changing much more rapidly than during the buildup. A new selection may now be started.

11.3 Learning space-feature patterns

Space-feature patterns are learned using the dynamics given by Equation 3.38 in Chapter 3. The extension of the equation to a three-dimensional space-



(a) Time course of the input and activation of the inhibitory node.



(b) Activation of the field driving the inhibitory node.

Figure 11.2: Exemplary time course of an inhibition node connected to a neural field.

feature pattern may be achieved trivially by appending another argument to the function, so that the dynamics of the pattern memory becomes

$$\tau_{\text{mem}} \dot{P}^{\text{mem}}_{F,l}(x,y,f,t) = -\left(P^{\text{mem}}_{F,l}(x,y,f,t) - P^{\text{sc}}_{\text{bu},F}(x,y,f,t)\right)\ b_{\text{lrn}}(t)\ p_l(t), \tag{11.22}$$

where τ_{mem} is the timescale of the adaptation, $b_{\text{lrn}}(t) \in \{0,1\}$ enables and disables learning and $p_l(t)$ is the estimate for label l.

11.4 Evaluation methods

In this section, I describe the methods I use to evaluate space-feature pattern architectures. These methods are used for the evaluation of all architectures described in Chapter 12 and Chapter 13.

11.4.1 Training procedure

To train the space-feature pattern memory, I adopt the procedure described in Lomp et al. (submitted 2016). During training, the dynamics of the architecture is simulated. This simulation runs without interruption or direct (algorithmic) changes to the state of the neural dynamics. Object views are trained individually, in a fixed order. When a view is presented to the architecture, label information is provided by additive input to the layer one and two nodes corresponding to the desired label. Analogously, Gaussian-shaped inputs centered at the pose of the object in the training image are added to the layer one and two fields that represent pose. The system is allowed to relax to the cued label and pose for a fixed period of time. Learning is then enabled by setting $b_{\text{lrn}}(t)$ in Equation 11.22 to one. The weights are allowed to change for a fixed duration, after which training of the current view is considered complete and $b_{\text{lrn}}(t)$ is again set to zero. The procedure is then repeated with the next view, until all object views have been trained.

11.4.2 Recognition and testing procedure

For testing, I also adapt the procedure described by Lomp et al. (submitted 2016). As during training, the architecture's dynamics is simulated without interruption or direct (algorithmic) changes to the state of the neural dynamics. At the start of the evaluation, the task *recognize* described in Section 7.2 is enabled. This task remains active for the entire duration of the evaluation. For each test image, the appropriate region of interest (see Chapter 6) is set as input to the system. The system is allowed to relax, and

recognition is considered complete when the *recognition done* node becomes active (that is, when its output of the node exceeds a threshold). When the duration needed for relaxation exceeds a maximal duration, the trial is ended immediately and recorded as a *timeout*. In both cases, the activation and output of the label nodes and pose fields at the end of the trial are recorded for evaluation. Afterwards, the next image may be presented to the system.

The precise method for choosing the next training image depends on the database. For the tabletop database, an image is chosen randomly until each image is presented once. I average results from multiple runs through this database, so that each image is presented four times.

For the *transformed* dataset, an image is randomly chosen from the training images of the tabletop dataset on each trial. Before passing this image into the architecture, the image is randomly scaled, rotated and translated (translation is applied last so that scaling and rotation happen around the image center) as described in Section 6.2. The transformation parameters for each trial are recorded and may thus serve as ground truth for the evaluation, replacing the manual annotations in the tabletop dataset. After applying the transformation, a region of interest corresponding to the center of the image is chosen from the resulting image and input into the architecture.

11.4.3 Performance measures

To evaluate the system's performance, I also follow the methods described in Lomp et al. (submitted 2016). At t_{end}, the end time of the trial being evaluated, the recognized label, l^*, is the label with the highest output:

$$l^* = \arg\max_l \left\{ g\left(u_{2,l}(t_{\text{end}})\right) \right\}. \tag{11.23}$$

Labels are rank-ordered based on their output level. This yields a list, $(l_1, l_2, \ldots, l_n)$ with $l_i \neq l_j$ and $g\left(u_{2,l_i}(t_{\text{end}})\right) \geq g\left(u_{2,l_{i+1}}(t_{\text{end}})\right)$ $\forall i \in \{1, \ldots, n-1\}$. From this list, the rank of a label l_i is determined as

$$\text{rank}(l_i) = i. \tag{11.24}$$

For correct recognitions, the annotated label for the image, l^{C}, is equal to the recognized label, l^*, and thus $\text{rank}(l^{\text{C}}) = 1$. The second choice of the system has rank two, and so on.

To calculate pose errors, the recognized pose must first be read out. I do this by finding the location of maximal output. Thus, the estimated position is

$$\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}) = \arg\max_{(x,y)} \left\{ g\left(u_2(x, y, t_{\text{end}})\right) \right\}, \tag{11.25}$$

where $u_2(x, y, t_{\text{end}})$ is the activation of the layer two field of the position representation (see Section 3.2). The position error is the Euclidean distance between the annotated pose (or randomly chosen pose for the transformed dataset), $\mathbf{x}^{\text{C}} = (x^{\text{C}}, y^{\text{C}})$, and the estimated pose:

$$E_{\text{sh}}(\mathbf{x}^{\text{C}}, \tilde{\mathbf{x}}) = \sqrt{(x^{\text{C}} - \tilde{x})^2 + (y^{\text{C}} - \tilde{y})^2}. \tag{11.26}$$

The estimated orientation, $\tilde{\phi}$, is analogously the maximum of the output of the layer two field of the orientation representation:

$$\tilde{\phi} = \arg\max_{\phi} \{g\left(u_2(\phi, t_{\text{end}})\right)\}. \tag{11.27}$$

where $u_2(\phi, t_{\text{end}})$ is the activation of the layer two field representing shift (please recall that the different pose fields are denoted by the same activation variable and are only differentiated by their arguments; see Section 3.2). The pose error is the shortest distance to the annotated (or, in the case of the transformed dataset, randomly chosen) orientation, ϕ^{C},

$$E_{\text{rot}}(\phi^{\text{C}}, \tilde{\phi}) = \begin{cases} |\tilde{\phi} - \phi^{\text{C}}| : & |\tilde{\phi} - \phi^{\text{C}}| < \pi \\ 2\pi - |\tilde{\phi} - \phi^{\text{C}}| : & \text{otherwise.} \end{cases} \tag{11.28}$$

The estimated scale is calculated as

$$\tilde{s} = f(\arg\max_{\rho}\{g\left(u_2(\rho, t_{\text{end}})\right)\}), \tag{11.29}$$

where f is a function that transforms the log-polar scale argument back to a linear scale factor. The pose error is the difference between the annotated scale, s^{C}, and the estimated one:

$$E_{\text{sc}}(s^{\text{C}}, \tilde{s}) = \left|s^{\text{C}} - \tilde{s}\right|. \tag{11.30}$$

Chapter 12

Space-edge patterns

In this chapter, I describe how space-edge patterns—space-feature patterns that are based on edge orientations[1]—may be extracted and used for the pose-transformation and matching architecture presented in the previous chapter.

I begin this chapter with a formal description of the pattern extraction. Edge orientations are locally aggregated into histograms. This is similar to the histograms used in the architecture described in Chapter 3, however, histograms are here extracted from much smaller regions. The size of these regions must reflect the scale estimated for the object, which implies that patterns are extracted on multiple scales.

Histogram calculation is computationally costly, and reducing the number of histograms that need to be extracted is therefore desirable. I achieve this by using keypoints as described in Chapter 5 which identify regions where edges of multiple orientations meet. Locations far from keypoints may thus be dominated by a single edge orientation, so that the histograms can be approximated by a single entry at the gradient orientation.

After formalizing pattern extraction, I describe the pose-transformation and matching approach, which mostly follows the one described in Chapter 11. An additional stage accounts for the rotation of the edge orientations in the feature space, much like the rotation of the edge orientation histograms described in Chapter 3. The resulting structure is shown in Figure 12.1. I also address issues arising from the coarse sampling of the edge orientations (which is, again, chosen due to computational constraints).

In the final sections of this chapter, I evaluate the system in different configurations. I consider the space-edge patterns to be a more biologically plausible representation for object views than the histograms used in the architecture presented by Faubel and Schöner (2009, 2010) and Lomp et al.

[1]Technically, these should be called space-edge orientation patterns, which I shorten to space-edge patterns for readability.

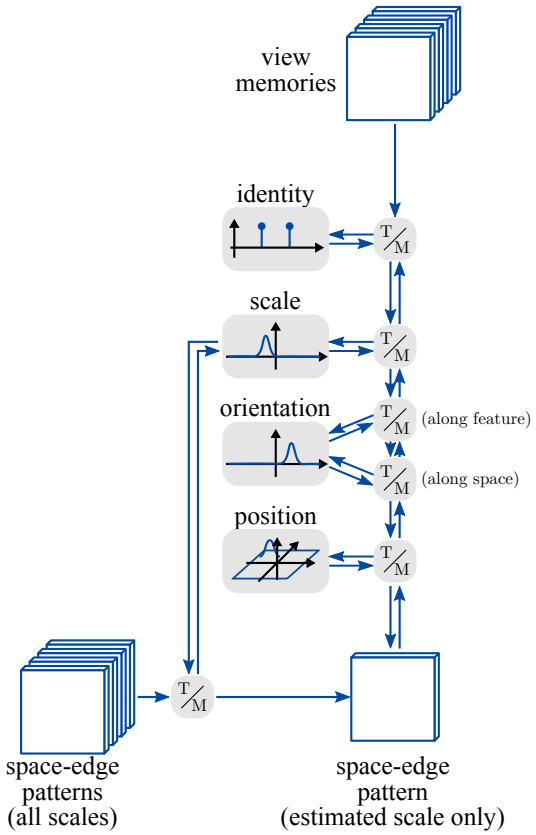


Figure 12.1: An overview of the space-edge pattern architecture. As before, boxes marked "T/M" stand for pose-transformation and matching steps.

(submitted 2016) because each entry in a space-edge pattern codes for the strength of an edge of a certain orientation at a specific retinal position, much like the simple cells described by Hubel and Wiesel (1962, 1968). I therefore use the pose-transformation and matching architecture developed in this chapter to characterize the system's link to biological vision.

12.1 Pattern extraction

To extract a space-edge pattern, the black channel (Y from the YUV color space) of the input image is first blurred by applying a Gaussian filter. This reduces the impact of high-frequency noise. Edge orientations are approximated by the image gradient, which is calculated using a Sobel filter. The result of this filtering is (an approximation of) the gradient's orientation, $\Theta : \mathbb{R}^2 \times \mathbb{R}^+ \to [0, \pi)$, and magnitude, $M : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}$.

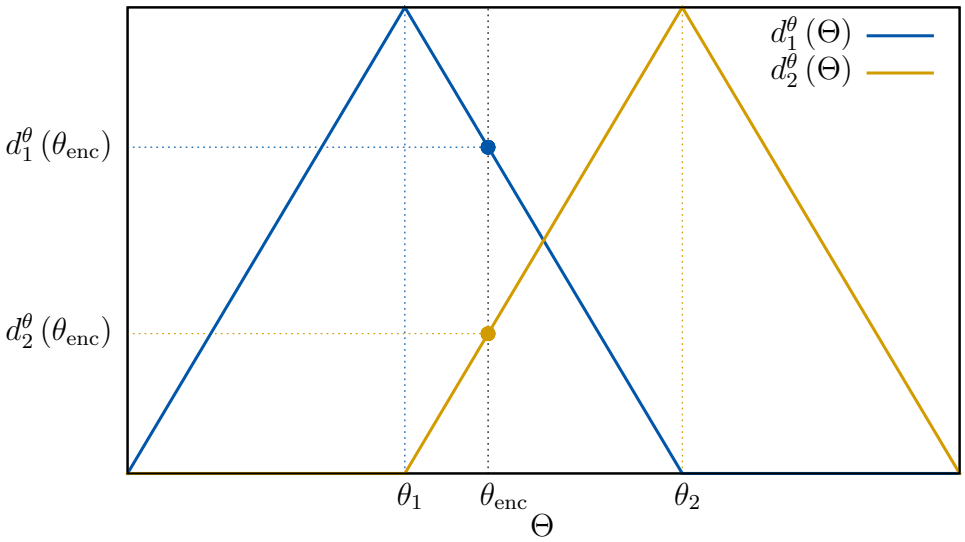


Figure 12.2: Illustration of the orientation subsampling function, $d_j^\theta(\Theta)$, for two orientation samples, θ_1 and θ_2. The output of the functions is illustrated for a value being encoded, θ_{enc}.

Keypoints are then extracted from the image using the approach described in Chapter 5, and the gradient orientations around them are aggregated into localized histograms. For each keypoint, i, located at image position $\mathbf{x}_i$ with scale σ_i, a histogram

$$h_i^{\text{edge,kp}}(\theta_j, t) = \iint G_{\sigma_i, \mathbf{x}_i}(x, y)\ d_j^\theta(\Theta(x, y, t))\ dx\ dy \tag{12.1}$$

is calculated, where $G_{\sigma_i, \mathbf{x}_i}(x, y)$ is a Gaussian function centered on the keypoint with width proportional to the scale of the keypoint (a total of ten different scales are used in the implementation). Note that I write down the equation using evenly spaced discrete gradient orientations, $\theta_j = j\ \Delta\theta$,[2] to indicate that the orientation dimension is sampled very coarsely (in practice, by eight different orientations). This reduces the computational complexity, but has nontrivial implications for the connectivity in the architecture. One such implication is the assignment of the (finely sampled) gradient orientation to one of the discrete orientations by the sampling function

$$d_j^\theta(\Theta) = \begin{cases} \frac{\Theta - \theta_{j-1}}{\Delta\theta} : & \theta_{j-1} \leq \Theta < \theta_j \\ 1 - \frac{\Theta - \theta_j}{\Delta\theta} : & \theta_j \leq \Theta < \theta_{j+1} \\ 0 : & \text{otherwise,} \end{cases} . \tag{12.2}$$

[2]The indices of this sampling are meant to be cyclic within the set of sampled orientations, that is, $\theta_{j-1} = \theta_{j-1 \bmod n}$, where n is the number of sampling points.

This sampling function defines a linear weight between zero and one for orientations based on their distance from sampling point j using the distance between two samples, $\Delta\theta$. This function is inspired by the idea of tuning curves (this is the inverse of the approach used for lifting coarsely sampled input to a finely sampled space described in Bicho et al., 2000; Schöner et al., 2015a). Due to the shape of this function (see Figure 12.2), I refer to it as a *simplified tuning curve*. The function is designed to preserve, up to a point, information on the sampled orientation that would be lost with other approaches such as nearest neighbor sampling.

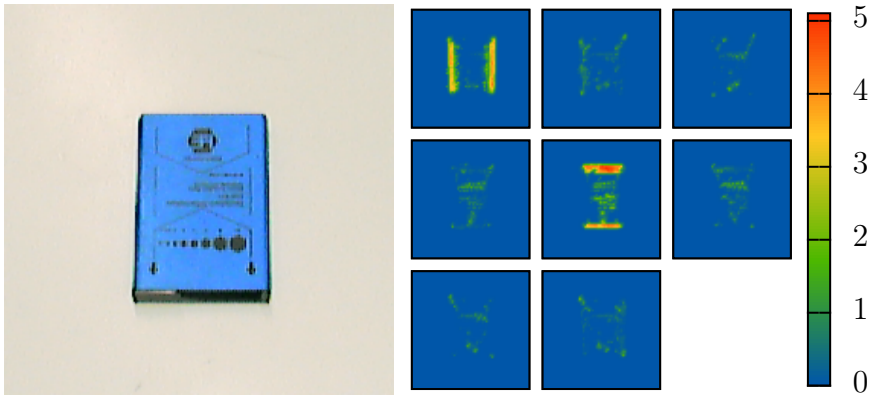
As an example, consider the output of the tuning curve for a single orientation to be encoded, θ_{enc}, and a set of sampling points, $d_j^\theta(\Theta)$. In nearest neighbor sampling, one of these would be assigned the value one, the others would be zero. With the simplified tuning curve sampling from above, the two closest bins, θ_j and θ_{j-1}, would both have nonzero values, as shown in Figure 12.2. The figure also demonstrates that a single sampling value, $d_j^\theta(\Theta)$, is not sufficient to find the encoded value, as there are two intersections with the tuning curve, and it is thus ambiguous whether the value is lower or higher than the sampling point. However, knowledge of the value for the other sampling point disambiguates this, making perfect reconstruction of the encoded orientation possible in this scenario. This is, essentially, population coding.

This sampling function is designed so that it can be computed efficiently because encoding a single value requires only to determine the two neighboring sampling points. With $v = \frac{\Theta - \theta_j}{\Delta\theta}$, the value for the bins is v for the lower bin and $1 - v$ for the upper bin, while all other bins have the value zero.

In principle, localized gradient orientation histograms can be extracted for each image location. However, this extraction is computationally costly, even with the coarse sampling. Thus, I make a simplifying assumption that image locations far from keypoints are dominated by a single edge orientation and thus have monomodal histograms. The feature pattern at these locations is thus approximated by

$$g^{\text{edge}}(x, y, \theta_j, t) = d_j^\theta(\Theta(x, y, t)), \tag{12.3}$$

and the final space-edge pattern is obtained by entering the histograms at keypoints and the approximated patterns far from keypoints into a single



(a) Input image. (b) Extracted space-edge pattern.

Figure 12.3: An example for a space-edge pattern. (a) shows the input image. (b) shows the extracted space-edge pattern. Each square in (b) stands for a different range of edge orientations (starting with 0, in steps of 22.5° from left to right, top to bottom).

---

structure,

$$P^{\text{edge}}_{\check{\sigma}_i}(x, y, \theta_j, t) = \sum_{k:\sigma_k=\check{\sigma}_i} \left[G_{\sigma_k} * h^{\text{edge,kp}}_k\right](x, y, \theta_j, t) + \sigma_+ \left(\left(\left[G_{\check{\sigma}_i} * g^{\text{edge}}\right](x, y, \theta_j, t) \right) - \sigma_{\text{abs}} \left(c^{\text{kp}}_{\check{\sigma}_i}(x, y) - \frac{1}{2} \right) \right), \tag{12.4}$$

where $\check{\sigma}_i$ are the scales on which keypoints are extracted, and σ_k is the scale of keypoint k. The function

$$c^{\text{kp}}_{\sigma_i}(x, y) = \sum_{k:\sigma_k=\sigma_i} G_{\sigma_i, \mathbf{x}_k}(x, y)\,. \tag{12.5}$$

formalizes the closeness of an image location to a keypoint. Where its value is zero, the values of the approximation in Equation 12.3 are used for Equation 12.4. Where it is nonzero, the input from Equation 12.3 becomes proportionally smaller in favor of the edge orientation histograms extracted around the keypoint locations. An example for a space-edge pattern is shown in Figure 12.3.

12.1.1 Input scale selection

To provide input to the pose-transformation and matching architecture, the space-edge patterns extracted on different scales are reduced to a single pattern by superposing them according to

$$P_{\text{edge}}(x, y, \theta_i, t) = \sum_i d_i^{\text{sc}}(t) P_{\text{edge}}^{\sigma_i}(x, y, \theta_i, t), \tag{12.6}$$

where d_i^{sc} samples the scale estimate, p_{sc}, according to

$$d_i^{\text{sc}}(t) = \int_{m_s(\sigma_{i-1})}^{m_s(\sigma_{i+1})} p_{\text{sc}}(\rho, t) d_i^{\text{tun}}(\rho)\ d\rho. \tag{12.7}$$

The function $m_s(\cdot)$ maps from the domain of the scale estimate to the domain of the input scales (details depend on the implementation and are given in Appendix C). The sampling function, $d_i^{\text{tun}}(\sigma)$, is defined analogous to the simplified tuning function (see Equation 12.2):

$$d_i^{\text{tun}}(\sigma) = \begin{cases} \frac{\sigma - \sigma_{i-1}}{\Delta\sigma} : & \sigma_{i-1} \leq \sigma < \sigma_i \\ 1 - \frac{\sigma - \sigma_i}{\Delta\sigma} : & \sigma_i \leq \sigma < \sigma_{i+1} \\ 0 : & \text{otherwise.} \end{cases} \tag{12.8}$$

12.2 Pose-transformation and matching architecture

Recall that edge histograms have to be shifted along the edge orientation dimension to account for rotation in the pose transformations (see Section 3.3). An analogous transformation must be applied for space-edge patterns. This implies additional transformation and matching steps in the architecture described in Chapter 11 that are added between the transformation by the orientation estimate and the scaling transformation.

In order to apply this additional transformation, the continuous orientation estimate must be sampled to match the sampling of the gradient orientations. The first step in this sampling is mapping the estimate from the larger interval, $[-\pi, \pi)$, to the smaller interval of possible edge orientations, $\theta \in [0, \pi)$ according to

$$p_{\text{edge}}^{\text{rot}}(\theta, t) = p_{\text{rot}}(\theta, t) + p_{\text{rot}}(\theta - \pi, t). \tag{12.9}$$

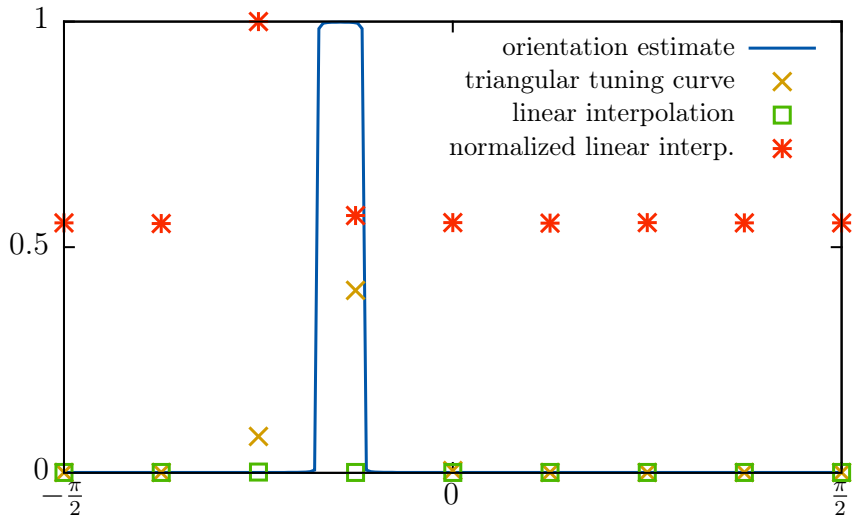


Figure 12.4: Effects of subsampling an orientation estimate with linear interpolation and simplified tuning curves. The result of linear interpolation normalized to its maximum value is also shown to make its shape more evident.

Next, this remapped orientation estimate is sampled at the discrete edge orientations using the simplified tuning curve defined in Equation 12.2. Thus, the resampled orientation estimate is

$$\tilde{p}_l^{\text{rot}}(t) = \frac{1}{N\Delta\theta} \int_{\theta_{l-1}}^{\theta_{l+1}} d_l^{\theta}(\theta)\, p_{\text{edge}}^{\text{rot}}(\theta, t)\; d\theta, \tag{12.10}$$

where $0 \leq l < N$ indexes the discrete edge orientations and $\Delta\theta$ is again the distance between two sampling points of the gradient orientation.

This last step in the sampling is necessary because only a small number of sampling points is used for the edge orientations. Standard sampling approaches, for example, linear interpolation, are problematic at such low resolutions because the narrow peaks in the orientation estimate may be lost in the subsampled estimate. The proposed sampling approach addresses this and better preserves peak positions, as illustrated in Figure 12.4: the result of linearly interpolating a finely sampled orientation estimate down to a coarsely sampled estimate is nearly zero everywhere. Normalizing this result using the maximum value reveals that near the peak position, values are still higher than the surrounding ones. However, this difference is small, and the surrounding values do not reflect those in the original orientation

estimate. When using this subsampled estimate in pose transformations, the result would thus contain all orientations and not reflect the specificity of the original orientation estimate. The simplified tuning curve, on the other hand, does not have these issues. Its value is significantly different from zero close to the rotation estimate. A second sampling point is above zero as well and expresses that the rotation estimate does not lie precisely on the sampling point. All other entries, however, are close to zero and thus match the original scale estimate. Transforming by this estimate would thus contain only orientations in a range matching the active region in the original estimate, and those orientations closer to it are weighted more strongly, so that the subsampled estimate reflects the original estimate more closely than the linearly interpolated version. However, sampling with the simplified tuning curve does not preserve the amplitude of the original orientation estimate (see Figure 12.4), even though Equation 12.10 contains a normalization term. This is not an issue for the architecture, as the transformed pattern is normalized before it is used for matching.

The subsampled orientation estimate is applied as a transformation along the feature dimension the bottom-up path. This transformation directly follows the spatial rotation (Equation 11.3; see also Figure 12.1, transformations labeled 'along feature' and 'along space') and is given by

$$P^{\text{frot}}_{\text{bu,edge}}(x, y, \theta_j, t) = \sum_k \tilde{p}^{\text{rot}}_{j-k}(t)\ P^{\text{rot}}_{\text{bu,edge}}(x, y, \theta_k, t), \tag{12.11}$$

where k iterates over the edge orientations, and $\tilde{p}^{\text{rot}}_{j-k}$ is the subsampled orientation estimate.[3] The resulting transformed pattern, $P^{\text{frot}}_{\text{bu,edge}}(x, y, \theta_j, t)$ replaces $P^{\text{rot}}_{\text{bu,edge}}(x, y, \theta_k, t)$ in the transformation by the scale estimate, Equation 11.4.

A corresponding inverse transformation along the feature dimension is inserted in the top-down path. This transformation operates on the scaled top-down superposition (Equation 11.12) and provides input to the spatial rotation (Equation 11.14). It is calculated as

$$P^{\text{frot}}_{\text{td,edge}}(x, y, \theta_j, t) = \sum_k \tilde{p}^{\text{rot,inv}}_{j-k}(t)\ P^{\text{sc}}_{\text{td,edge}}(x, y, \theta_k, t), \tag{12.12}$$

where $\tilde{p}^{\text{rot,inv}}_l$ is the inverse rotation parameter estimate (as defined in Chapter 3), sampled as described by Equation 12.10.

The space-edge patterns also provide an additional match value for the

[3] Again, the index $j - k$ is cyclic in the number of sampling points.

orientation of the object according to

$$\text{match}_{\text{rot}}^{\text{edge},\theta}(\theta_i, t) = \sum_j \iint \hat{P}_{\text{bu,edge}}^{\text{rot}}(\rho, \phi, \theta_i - \theta_j, t)\ \hat{P}_{\text{td,edge}}^{\text{sc}}(\rho, \phi, \theta_i, t)\ d\rho\ d\phi. \tag{12.13}$$

Inspired by the ideas from Bicho et al. (2000) and Schöner et al. (2015a), these discrete values are lifted to the full sampling of the orientation estimate by determining the amplitude of Gaussians centered on the sampled gradient orientations,

$$\text{match}_{\text{rot}}^{\text{edge},\tilde{\theta}}(\phi, t) = \sum_i G_{\sigma_\theta}(\theta_i - \phi, t)\ \text{match}_{\text{rot}}^{\text{edge},\theta}(\theta_i, t), \tag{12.14}$$

where the width of the Gaussians, σ_θ is proportional to $\Delta\theta$. These Gaussians, in turn, provide input to the first layer field of the orientation representation (see Section 3.2.1).[4]

Space-edge patterns may provide additional scale match values because they are extracted on multiple scales. These values are obtained by comparing the top-down pattern, $P_{\text{td},F}^{\text{sh}}(x, y, \theta_j)$, with the input pattern for the corresponding scale, σ_i:

$$\text{match}_{\text{sc}}^{F,\sigma_i}(t) = \sum_j \iint \hat{P}_F^{\sigma_i}(x, y, \theta_j, t)\ \hat{P}_{\text{td},F}^{\text{sh}}(x, y, \theta_j, t)\ dx\ dy. \tag{12.15}$$

The discrete match values are lifted to the full range of scale estimates using the same idea described above, that is, by summing Gaussians centered around the scale values,

$$\text{match}_{\text{sc}}^{F,\text{I}}(\rho, t) = \sum_i \text{match}_{\text{sc}}^{F,\sigma_i}(t)\ G_{\sigma_\rho}(m_s(\sigma_i) - \rho), \tag{12.16}$$

where σ_ρ scales the range of the contribution of each discrete match value and $m_s(\cdot)$ maps from the domain of the scale estimate to an input scale space as described in Appendix C. The match value thus obtained provides an additional input to the layer one scale representation field.

12.3 Performance evaluation

I evaluate the architecture presented in this chapter in two steps. First, I take a closer look at the performance in terms of recognition rates and pose

[4] In practice, each edge orientation drives the amplitude of two Gaussians, one at the edge orientation θ_i, and one at $\theta_i + \pi$, to lift the estimate to the full range covered by the orientation representation. This is left out of the equation for clarity.

Table 12.1: Performance for different databases and architectures. Results on the histogram approach are reprinted from Lomp et al. (submitted 2016).

architecture	dataset	recog. rate	avg. rank	avg. pose errors		
				pos.	ori.	scale
histograms	tabletop	87.2%	1.2	13.5 px	14.0°	—
histograms, no color	tabletop	21.3%	11.2	22.6 px	16.8°	—
$\frac{1}{4}$ res.	tabletop	57.9%	4.4	11.0 px	81.5°	0.09
$\frac{1}{2}$ res.	tabletop	62.8%	4.5	10.6 px	75.4°	0.08
$\frac{1}{4}$ res.	transformed	69.0%	4.0	7.4 px	35.5°	0.11
$\frac{1}{4}$ res.	COIL (first 30)	52.7%	6.3	—	—	—

errors to determine how well the object representation performs in a robotic context analogous to that used to evaluate the original histogram-based architecture (see Faubel and Schöner, 2009, 2010; Lomp et al., submitted 2016). Second, I investigate how the architecture relates to behavioral and electrophysiological data on the human and primate visual system.

Table 12.1 summarizes the performance obtained from experiments using different datasets (see Chapter 6) and different architectures. The 'histograms' architecture refers to the full histogram architecture described in Chapter 3. Results are reprinted from Lomp et al. (submitted 2016) for comparison. The 'histograms, no color' architecture refers to a variant of this architecture that uses all but the color histogram channels. The architectures labeled with '$\frac{1}{n}$ res.' refer to architectures that use space-edge patterns as described above. The fraction indicates the spatial resolution of the architecture: for $\frac{1}{2}$ resolution, the space-edge patterns have half the resolution of the region of interest provided to the architecture, that is, a size of $128 \times 128 \times 8$ pixels (where the first two numbers indicate the size in image space and the latter indicates the sampling of edge orientations). For $\frac{1}{4}$ resolution, the patterns have a size of $64 \times 64 \times 8$ pixels. It is worth noting that even though the architecture's performance is below that of the full histogram architecture for both resolutions, it does perform better than the histogram architecture when comparable input features are used. The histogram architecture even used a higher feature resolution (36 rather than 8 edge orientations), and performance would suffer if the feature resolution was further reduced (Lomp et al., submitted 2016).

Results for the COIL database were obtained from 6085 trials. The architecture was trained using only a single training view for each object. Given this low amount of training images, the relatively good performance is surprising. In part, this can be explained by symmetries with respect to rotation in depth present for some of the chosen objects (see Section 12.4.2).

The confusion matrix for the tabletop dataset obtained with the $\frac{1}{4}$ resolution architecture is shown in Figure 12.5. It shows that objects with similar shape, such as the red and blue box and the different screw drivers, tend to be confused. The explanations for other confusions are not as obvious. For example, the *hanuta* and the *tape dispenser* are often confused, even though the *hanuta* has a much more complex texture than the *tape dispenser*. However, these particular objects are relatively small and therefore take up only a small region of the test and training images. As a result, much of their detailed structure is lost due to sampling as well as the blurring caused by pose-transforming the input in the bottom-up path.

Other issues may be attributed to depth rotations. For example, the *shampoo* bottle is never recognized properly. This is at least in part caused by its upright placement in the test images, which leads to large variations of appearance due to depth rotation. However, the system also has difficulty recognizing it in the transformed dataset (see below), which eliminates depth rotations.

For the transformed database, results were obtained from 13436 trials. The confusion matrix is shown in Figure 12.6. The results are similar to the tabletop dataset, except that some objects are recognized correctly more often, for example, the *bit box* and the *yellow stapler*. Some of this improvement may be attributed to depth rotations as discussed above.

Figure 12.7 compares the position errors on the tabletop dataset measured for the different approaches. In all cases, the position errors are grouped around zero. Although the errors for the space-feature patterns are grouped around zero more closely, some outliers (up to 220 pixels) lead to a larger average error value. When only trials resulting in correct recognitions are averaged, all position errors are smaller than 35 pixels for the space-edge patterns whereas the range of errors for the histogram architecture remains close to the overall result. Increasing the spatial resolution leads to a slightly more pronounced grouping of the position errors around zero, and the average position error consequently decreases slightly (see Table 12.1). Position error histograms for the transformed dataset are similar to the half-resolution results and therefore are not shown here.

Figure 12.8 and Figure 12.9 compare the orientation errors for the different approaches on the tabletop and transformed dataset (for the transformed dataset, half-resolution results are not available because of the computational

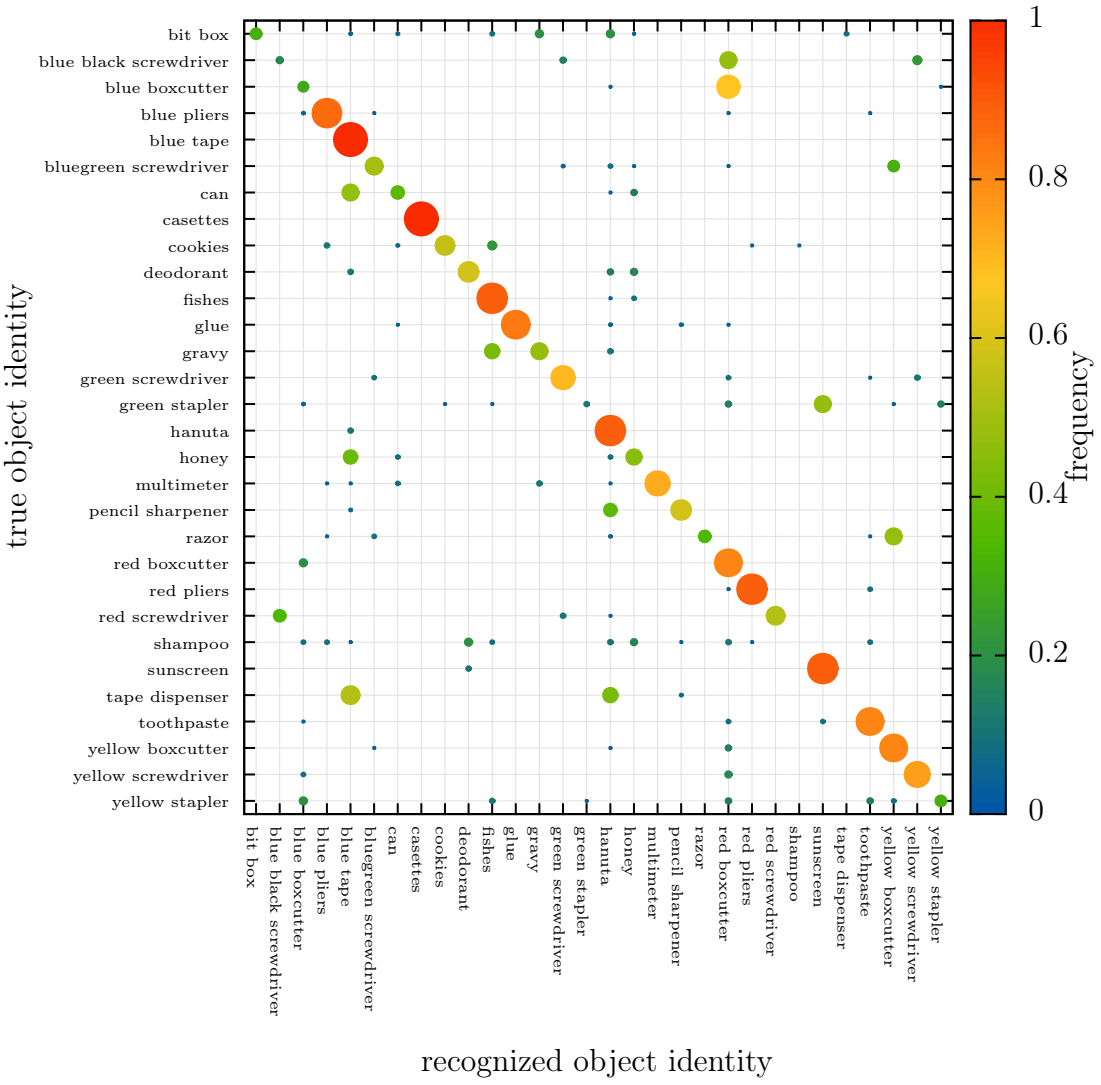


Figure 12.5: Confusion matrix for the tabletop dataset and $\frac{1}{4}$ resolution architecture. Dot size is proportional to the relative frequency.

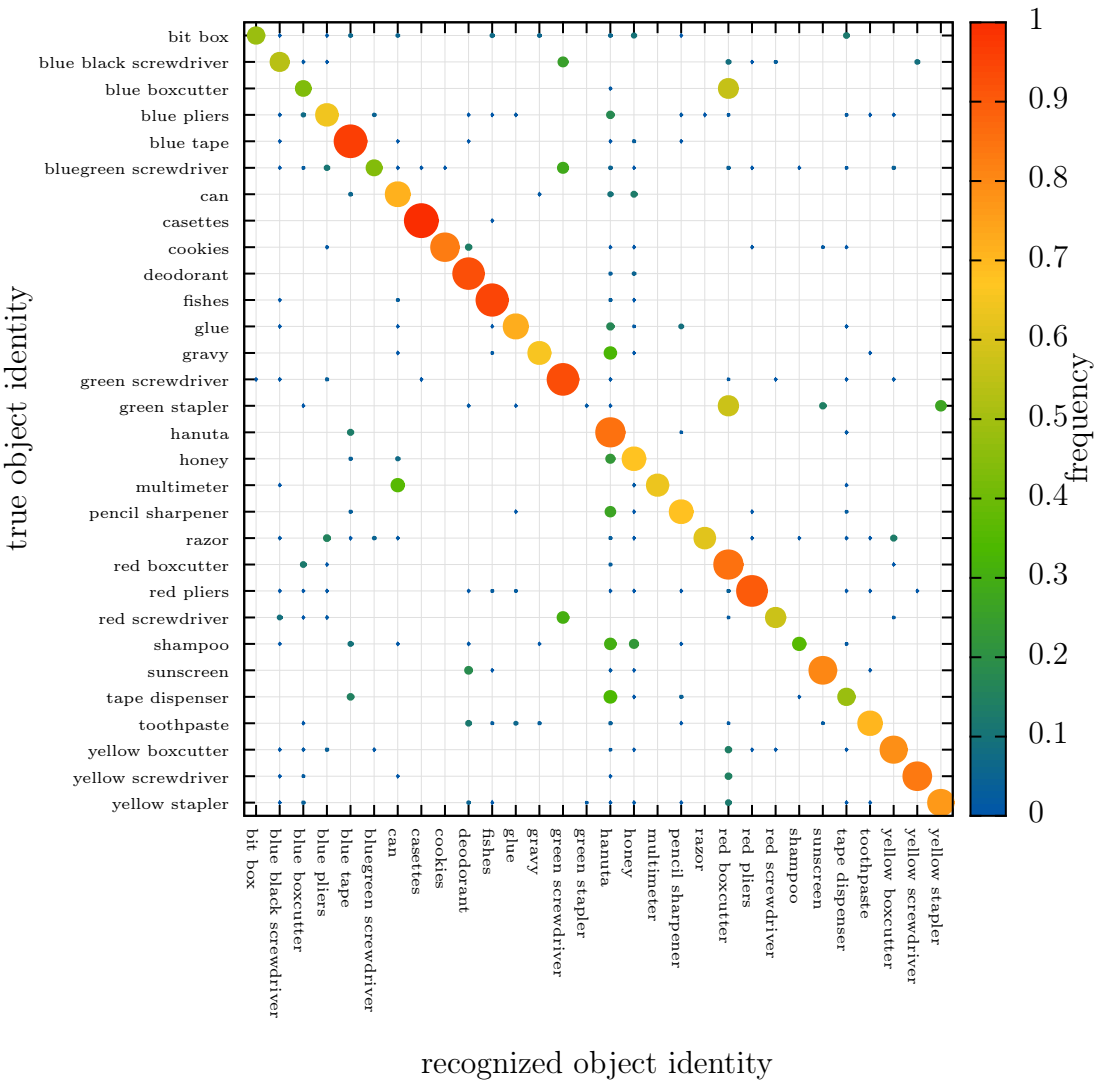
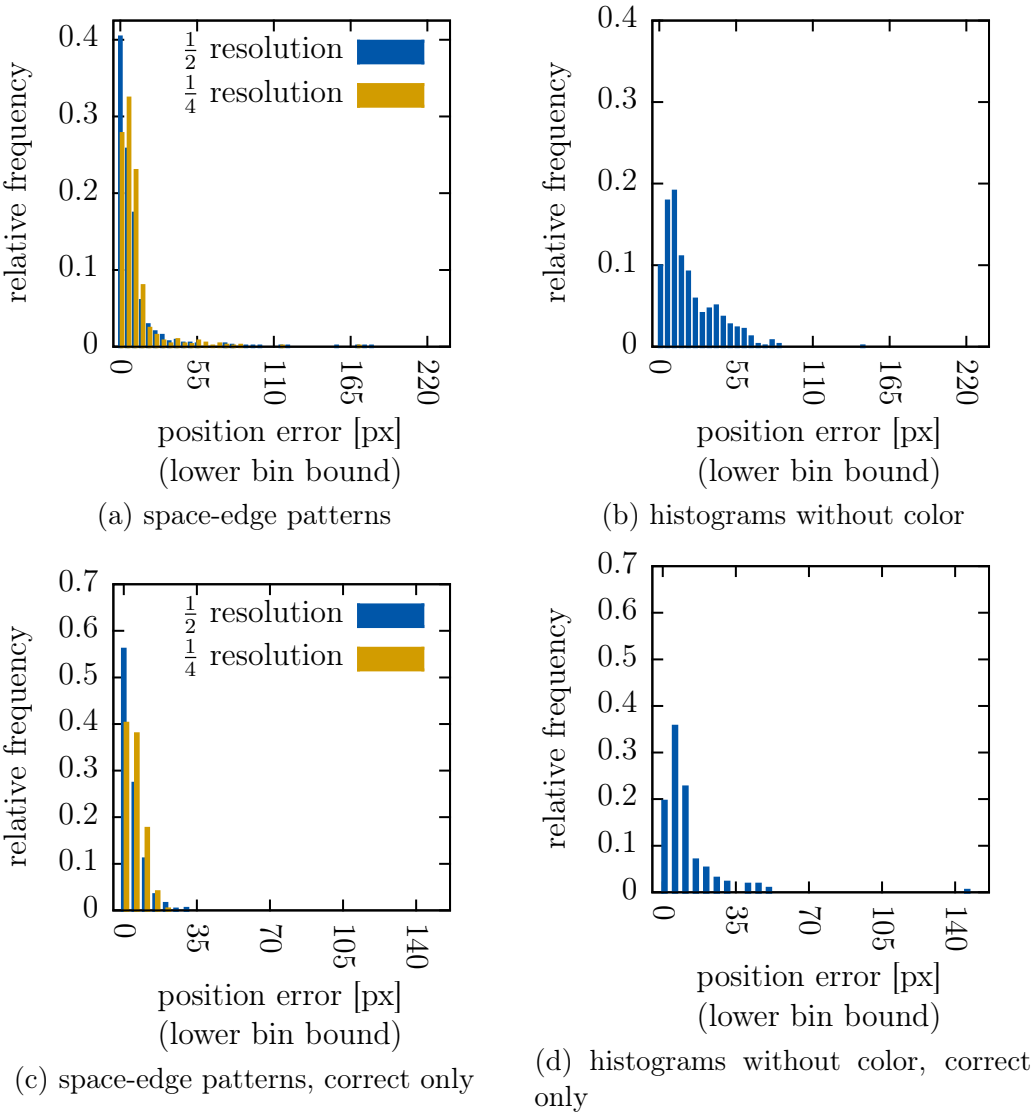


Figure 12.6: Confusion matrix for the transformed dataset obtained with the $\frac{1}{4}$ resolution architecture. The size of the dots is proportional to the relative frequency.



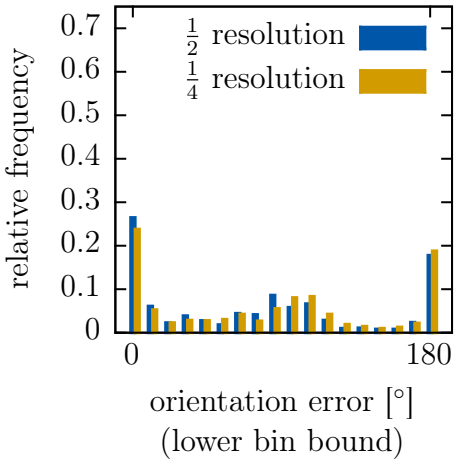
(a) space-edge patterns

(b) histograms without color

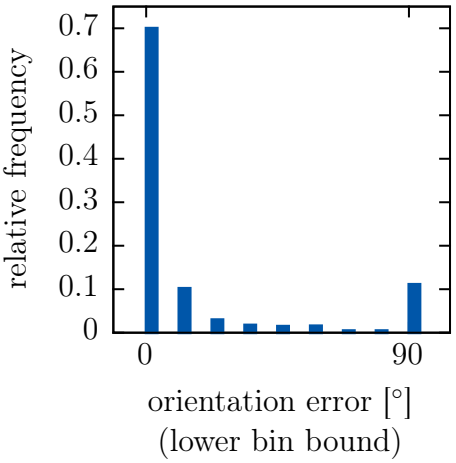
(c) space-edge patterns, correct only

(d) histograms without color, correct only

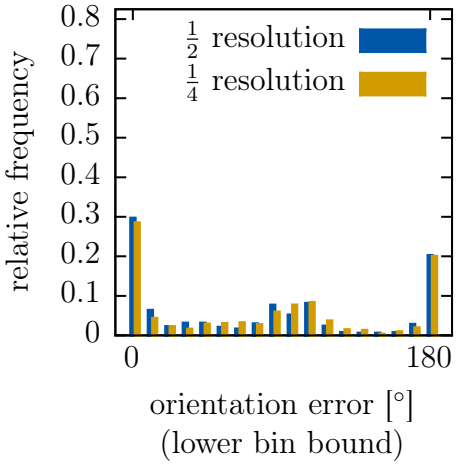
Figure 12.7: Histograms of the position errors on the tabletop database. The left column shows the histograms for space-edge patterns (half and quarter spatial resolution). The right column shows histograms for the histogram system without color. Histograms in the top row are computed over all recognitions. Histograms in the bottom row are computed from correct recognitions only. Note that the axis scaling differs between the top and bottom row.



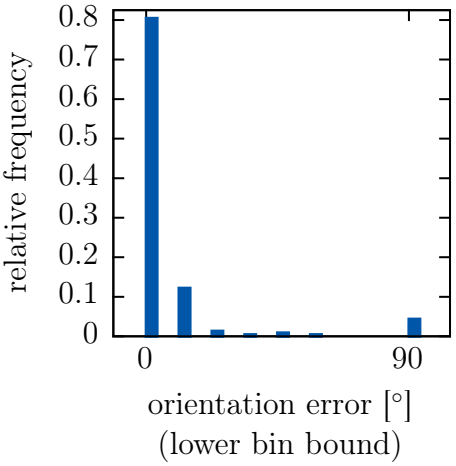
(a) space-edge patterns



(b) histograms without color

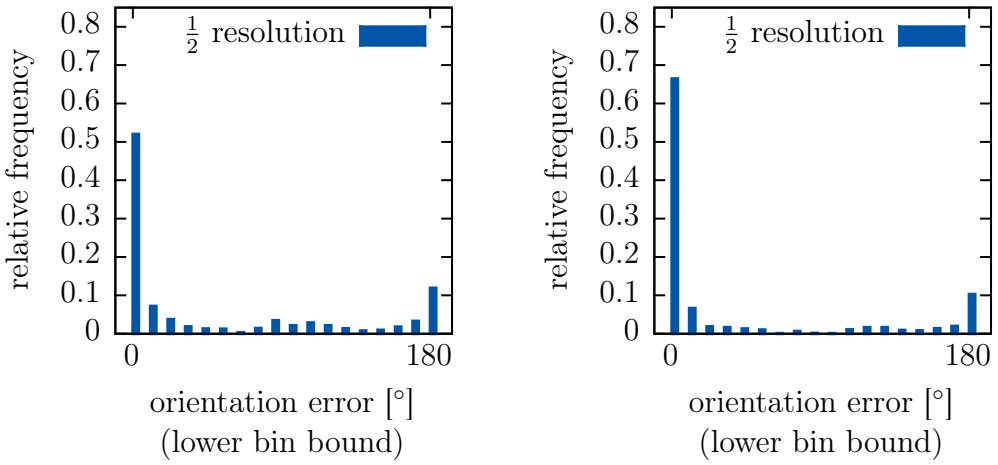


(c) space-edge patterns, correct only



(d) histograms without color, correct only

Figure 12.8: Histograms of the orientation errors on the tabletop database. The left column shows the histograms for space-edge patterns (half and quarter spatial resolution). The right column shows histograms for the histogram system without color. Histograms in the top row are computed over all recognitions. Histograms in the bottom row are computed from correct recognitions only.



(a) all recognitions

(b) correct recognitions only

Figure 12.9: Histograms of the orientation errors on the transformed database. As in Figure 12.8, the left column shows the histograms for space-edge patterns. The right column shows histograms computed over all recognitions. The histogram in the right column is computed from correct recognitions only.

---

time required to reach a sufficient number of trials). To an extent, the comparison between the histogram-based approach and the space-feature pattern approach is inadequate as both approaches estimate orientation over different ranges. Regardless, the figure shows that the space-edge pattern approach generates a more diverse distribution of orientation errors than the histogram approach. This likely has two reasons. First, the histogram approach samples edge orientations much more finely, thus making a more precise estimation of object orientation possible. Second, the orientation estimation in the space-feature pattern approach is influenced both by rotation of the edges as well as spatial rotation. The latter yields less precise estimates for rotation because the patterns are transformed by the shift and scale estimates before being matched, leading to blurring of the patterns being matched. This increases the range of possible rotational matches, allowing for a larger error to occur.

Moreover, the figure shows a tendency for orientation errors to group around 0°, 90° and 180°. These groupings reflect symmetries in the objects, as demonstrated in more detail by Figure 12.10, which shows that objects with symmetric shape such as *gravy* and *hanuta* (both close to a square shape) tend to induce exactly such orientation errors. On the other hand, objects such as the *bit box* which have no discernible "up"-direction in the

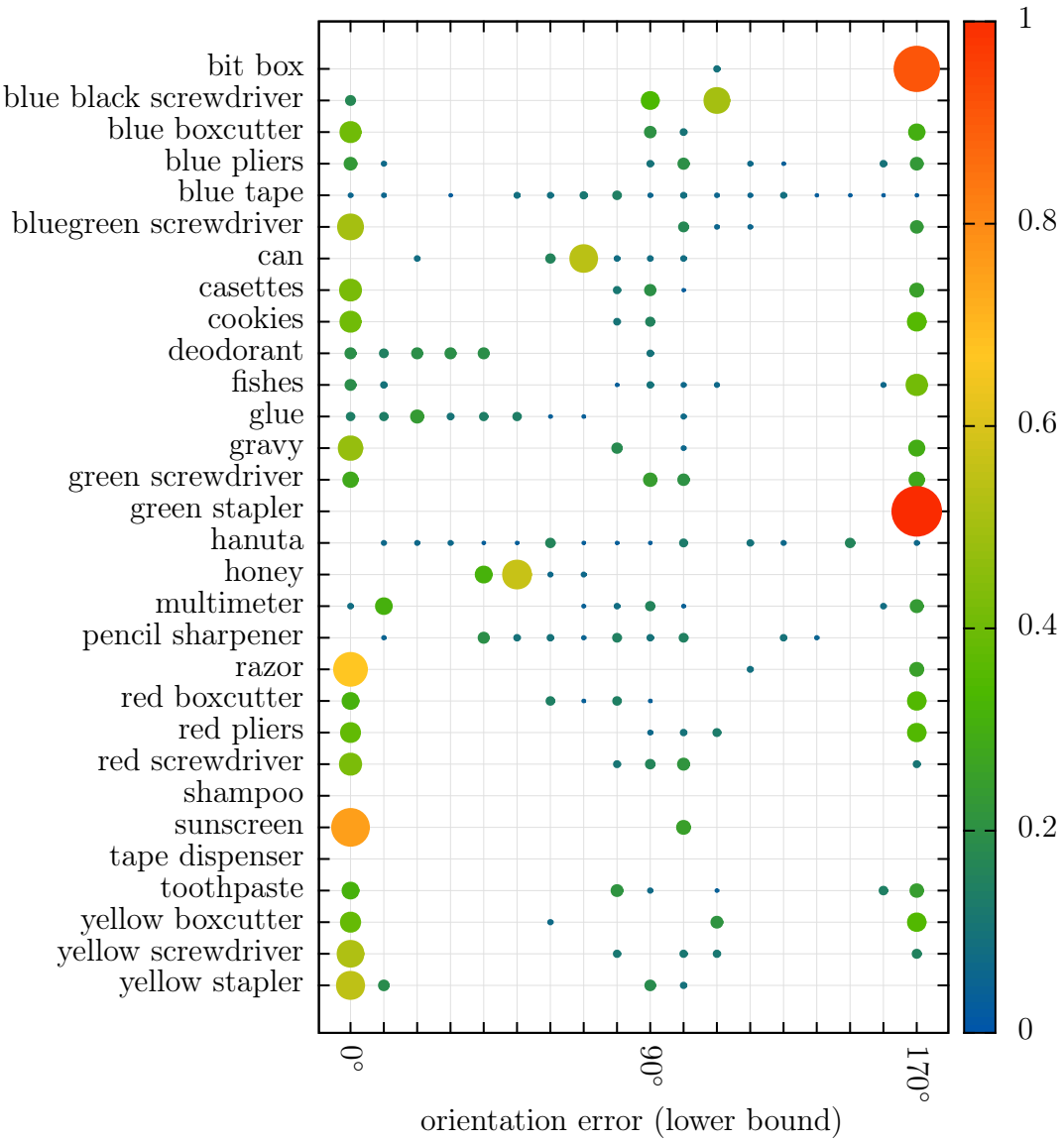


Figure 12.10: Per-object distribution of orientation errors on the tabletop dataset for $\frac{1}{4}$ resolution. The size and color of the dots indicates the relative frequency of the error.

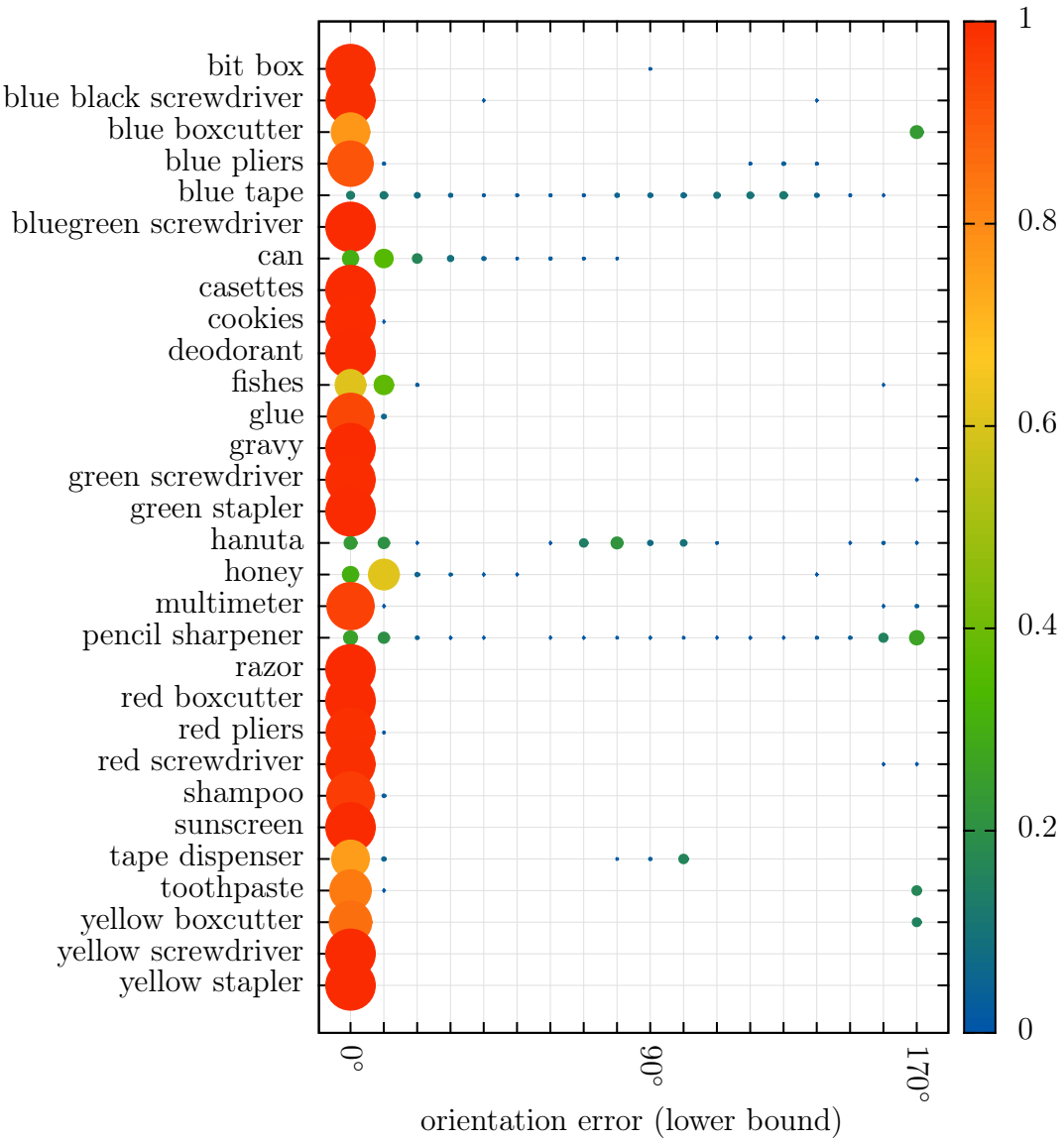
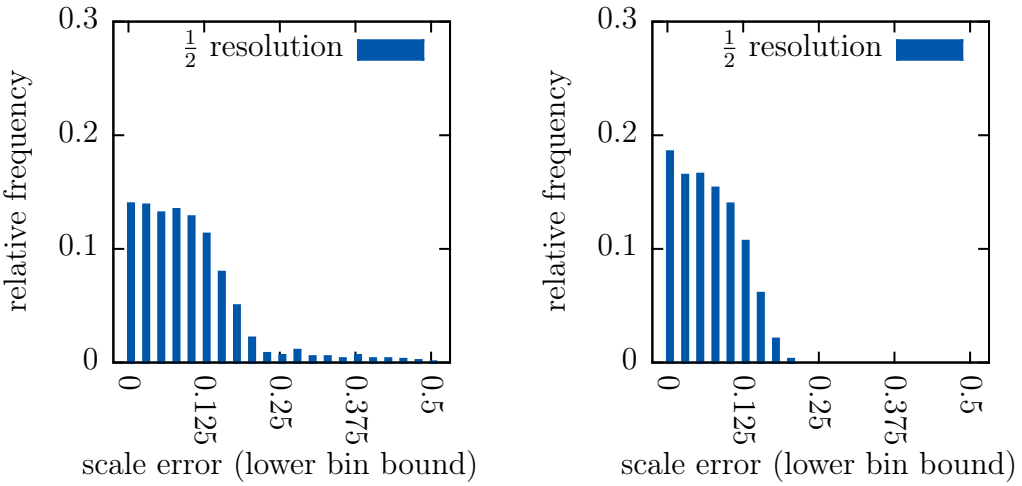


Figure 12.11: Per-object distribution of orientation errors on the transformed dataset for $\frac{1}{4}$ resolution. The size of the dots indicates the relative frequency of the error.



(a) all recognitions (b) correct recognitions only

Figure 12.12: Histograms of the scale errors on the transformed database. A small number of scale errors above 0.5 have been excluded from (a) for readability.

---

representation of the space-edge patterns frequently induce orientation errors of 180°. This is also apparent in Figure 12.11, which shows the distribution of orientation errors for all objects obtained with the transformed dataset. Orientation errors due to perspective transformations are excluded by the nature of the dataset, and effects from symmetry remain as the main explanation for these errors. The figure also shows another case that occurs for the *blue tape*. This is a round object with no clear markings indicating its orientation. Consequently, the architecture cannot properly estimate its orientation, and the resulting orientation errors are distributed over the entire space of possible errors, indicating a quasi-random choice.

Figure 12.12 shows histograms of the scale errors obtained for the space-edge pattern architecture on the transformed dataset (the tabletop dataset does not test scale systematically). The errors again group around zero, and most are smaller than 0.2. As for the orientation errors, this relatively large error margin likely stems from the blurring caused by the pose transformations. This allows for a larger deviation when matching the size of patterns, and hence results in larger scale errors. A comparison with the histogram architecture is not possible because it does not estimate scale.

12.4 Characterization of the system's behavior

Though many aspects of the histogram-based object recognition system are neurally plausible, its link to the human vision system has not been formally established. In the present section, I describe additional data that provides the basis for investigating this link. This data is based on the results I obtained for the space-edge pattern system described in the previous section. In Section 12.5, I relate the results to experimental data from the literature.

I first investigate how the object's pose (both inside and outside the image plane) affects the performance of the object recognition system. These experiments qualitatively reproduce studies on mental rotation (Shepard and Metzler, 1971; Tarr and Pinker, 1989) and experiments investigating recognition of wire stimuli in different orientations (for example, Bülthoff and Edelman, 1992).

I use two main measures for performance in these evaluations. Recognition performance measures the percentage of correct recognitions as described in Section 11.4.3. This measure is the inverse of the error rate measured in, for example, the experiments reported by Bülthoff and Edelman (1992). Another measure commonly used in experiments is response time. An exact version of such a measure is not available for the object recognition architecture because in humans, the response includes further processes such as pressing a button or producing a verbal response. I therefore only approximate response time by the time to convergence, that is, the time from the presentation of the image to the activation of the recognition done node (t_{end} in Section 11.4.3).

12.4.1 Effects of in-plane transformations

In the present subsection, I present an evaluation of the effects of translation, rotation and scaling in the image plane on the behavior of the object recognition system. Results are aggregated from the 13436 trials with the transformed tabletop dataset, using $\frac{1}{4}$ spatial resolution (see also Table 12.1).

12.4.1.1 Effects on recognition performance

In Figure 12.13, recognition performance for different object poses is shown (since the possible poses in the *transformed* dataset are taken from continuous value ranges, values are assigned to bins that are 5 pixels wide for translation, 10° wide for orientation, and factors of 0.025 wide for scale). Note that the last bin for position was only sampled sparsely because the position of the object was varied in a rectangle (uniformly random x and y offset) that

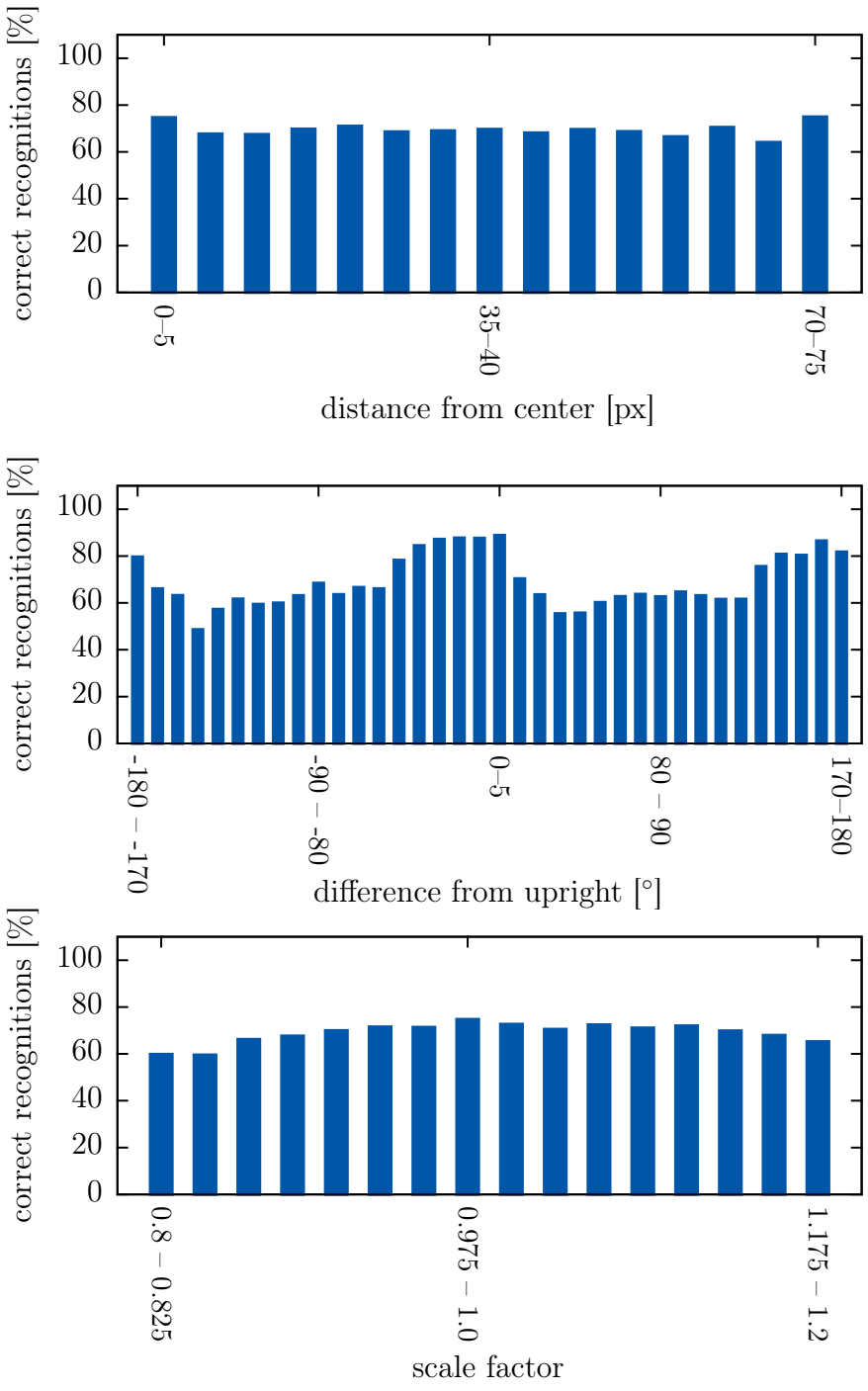


Figure 12.13: Percentage of correct recognitions for different deviations from the learned pose.

is here reduced to the distance from the rectangle's center. The last bin therefore contains the largest possible distances which have a lower chance of occurring. Therefore, only a small number of trials fall into this bin, so that the average is less precise than the averages in the other bins, possibly explaining its deviation from the other bins.

To determine whether there is a correlation between the pose parameters and recognition rate, I fitted a logistic regression model with three coefficients. The resulting model (intercept $\beta_0 = -1.6$; position coefficient, $\beta_{\text{sh}} = -0.0033$; orientation coefficient, $\beta_{\text{rot}} = -0.000\,80$; and, scale coefficient $\beta_{\text{sc}} = 2.2$) is highly significant ($\chi^2 = 30.1$, $p < 0.001$).

I also investigated the dependence of the distance (rather than the difference) to the learned pose by again fitting a logistic regression model with three coefficients. For the regression, orientations were mapped to the distance to the upright (learned) orientation, that is, $\phi' = |\phi|$ (where ϕ' is the mapped orientation and ϕ the original orientation in the range $[-\pi, \pi]$). Scales were similarly mapped as $s' = |1 - s|$ (where s' is the mapped scale and s is the original scale). Position was not remapped as it is already expressed as a distance from the learned position. The resulting coefficients (intercept $\beta_0 = 1.1$; position coefficient, $\beta_{\text{sh}} = -0.0032$; orientation coefficient, $\beta_{\text{rot}} = -0.0010$; and, scale coefficient $\beta_{\text{sc}} = -4.1$) are again highly significant ($\chi^2 = 25.6$, $p < 0.001$).

The probabilities resulting from the logistic models (when setting all but the variable being plotted to correspond to identity transformation) are illustrated in Figure 12.14. As the figure shows, the effects of position and orientation are very weak over the tested pose ranges (in both cases, the probability of correct recognition drops by only a few percent). The strongest effect occurs for scale, for which the predicted recognition rate drops by nearly 20% at the largest tested deviations from the scale of the training stimulus.

12.4.1.2 Effects on convergence times

In the experiments, a timeout ensured the termination of recognition trials (see Section 11.4.3). Trials in which this timeout was reached are excluded from the convergence time analysis because they skew the analysis towards pose-independent performance. Figure 12.15 shows the raw convergence times (also excluding cases where the timeout was reached) for correct and incorrect recognitions over the distance between the tested and learned position (for the following analysis of convergence time, no distinction is made between correct and incorrect recognitions).

To check the relation between convergence time and position (here measured as the distance from the position in which the object was learned), I

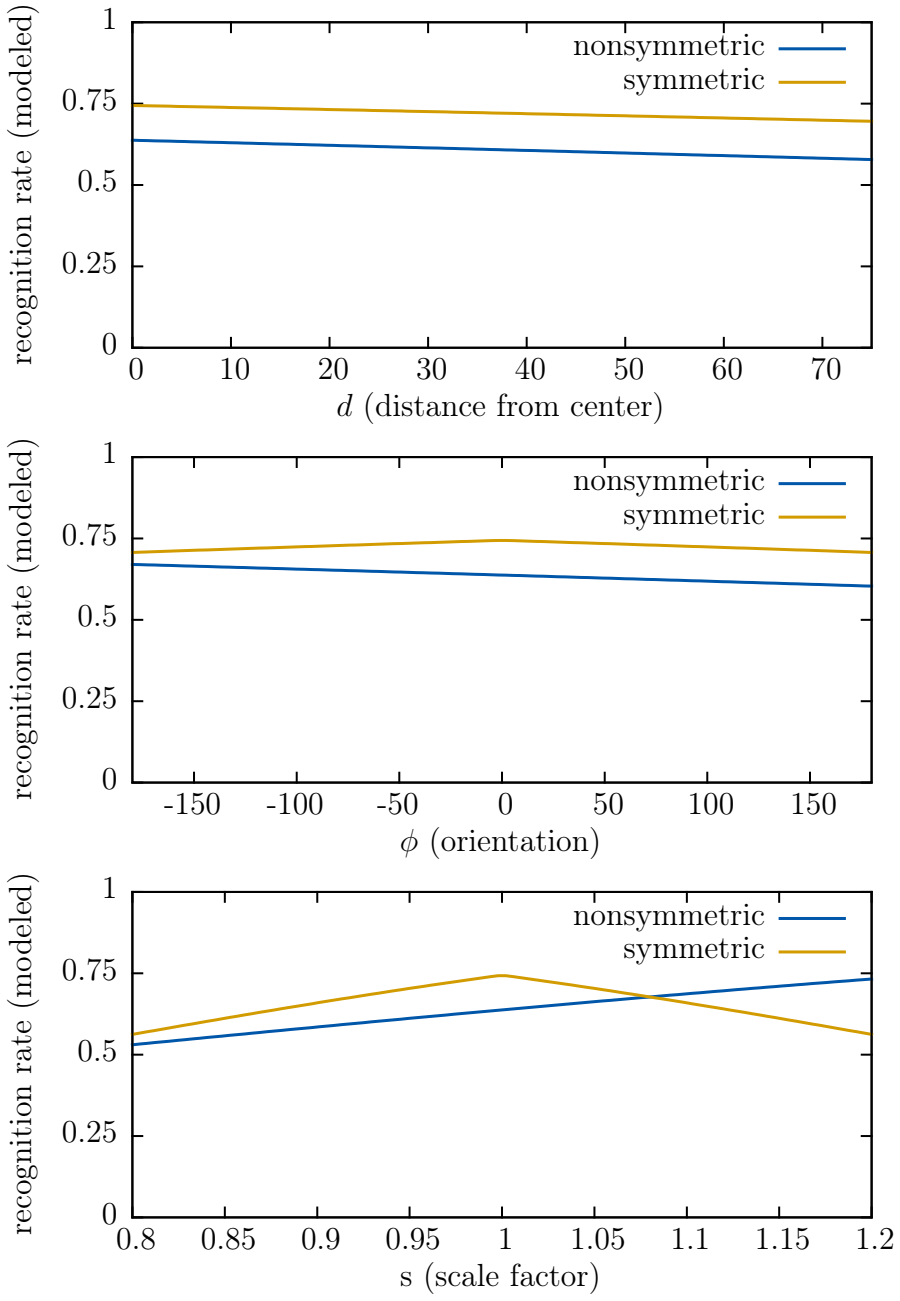


Figure 12.14: Probabilities of correct recognition predicted by a logistic regression model based on observed recognitions. Each figure shows the dependence of the modeled recognition rate on one of the parameters when all other parameters have been set to correspond to identity transformations ($d = 0$, $\phi = 0$ and $s = 1$, respectively).

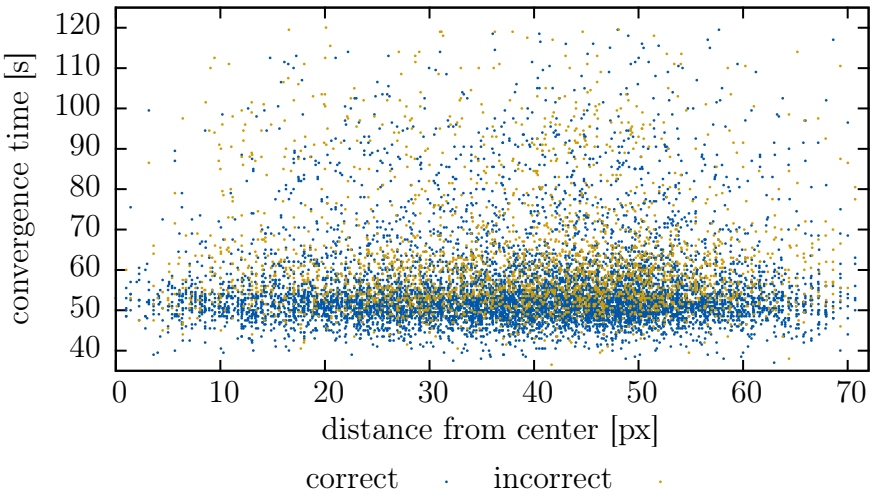


Figure 12.15: Raw convergence times plotted over the distance from the center of the image (the distance to the position of the object in the training image).

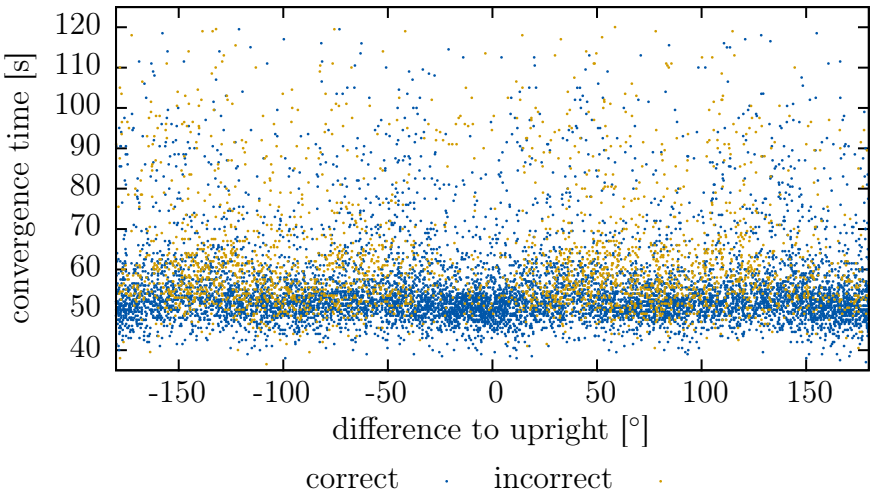
---

calculated Spearman's correlation coefficient. The resulting correlation coefficient, $\rho = 0.017$, is very weak. A two-sided t-test reveals that it is not significant ($t(11041) = 1.757$, $p > 0.05$; t statistic calculated with the method described on p. 640 of Press et al., 1992).

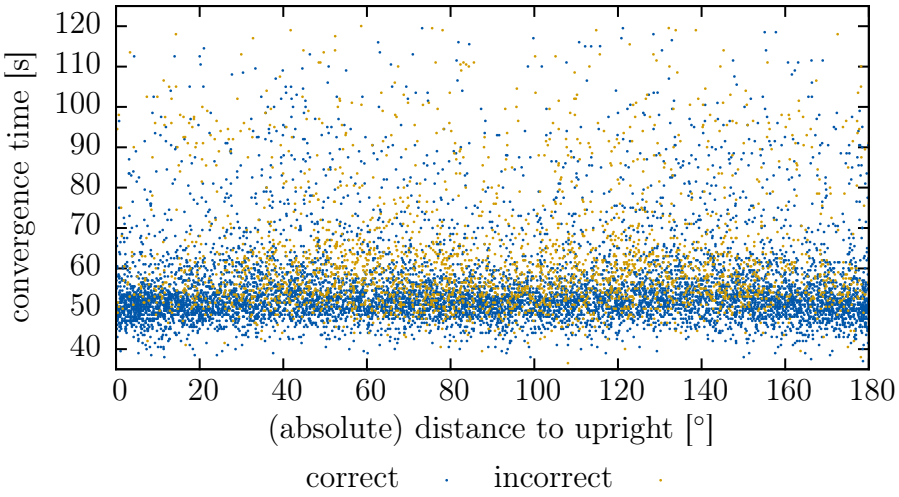
Figure 12.16 shows the raw convergence times (again excluding timed out cases) for correct and incorrect recognitions over the difference between tested and learned orientation. To check the relation between convergence time and difference between learned and tested orientation, I again calculated Spearman's correlation coefficient. The resulting coefficient is again very weak ($\rho = -0.045$), but is significant ($t(11041) = -4.683$, $p < 0.001$).

In mental rotation experiments (for example, Tarr and Pinker, 1989), rotation depends on the distance to the upright orientation. To perform an analogous test, I calculated Spearman's correlation coefficient using the distance to the learned orientation (see Section 12.4.1.1). The correlation coefficient is again very weak ($\rho = 0.038$) and significant ($t(11041) = 4.029$, $p < 0.001$).

Figure 12.17 shows the raw convergence times (excluding timed out cases) for correct and incorrect recognitions over the scale factor between the tested and learned object view. Spearman's correlation coefficient for response times vs. scale is very weak ($\rho = 0.01$) and is not significant ($t(11041) = 1.373$, $p > 0.05$).

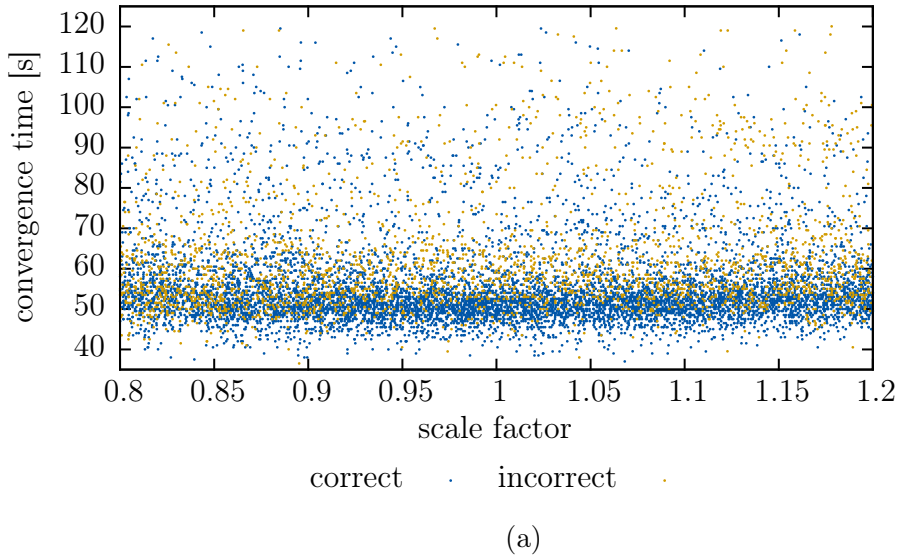


(a)

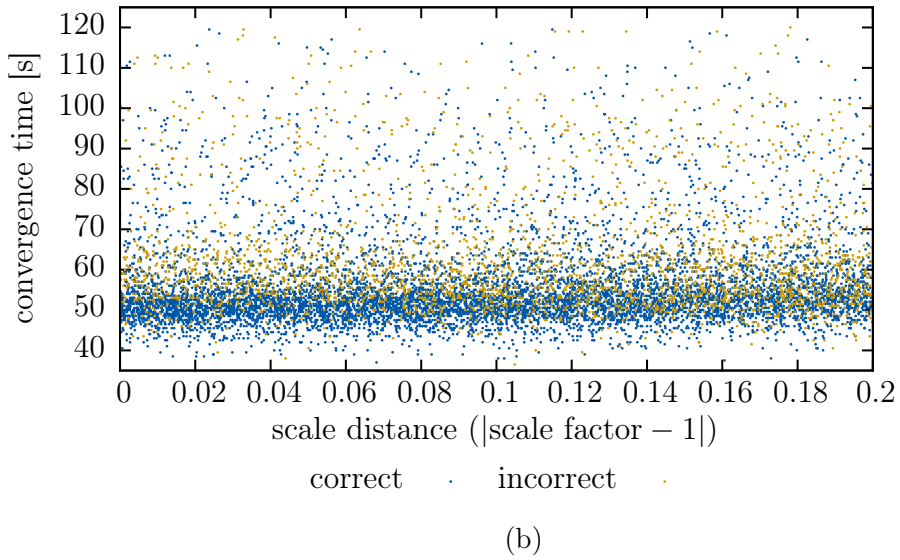


(b)

Figure 12.16: Convergence times over (a) the full range of orientations and (b) over the absolute value of the orientations, that is, the shortest distance to the orientation of the training image.



(a)



(b)

Figure 12.17: Convergence times over (a) the full range of scale factors and (b) over the distance from the scale of the training image.

In analogy to distance to the learned orientation, I also investigate the relation of convergence time to the distance to the learned scale (see Section 12.4.1.1). Spearman's correlation coefficient in this context is again very weak ($\rho = 0.11$), but is significant ($t(11041) = 11.125$, $p < 0.001$).

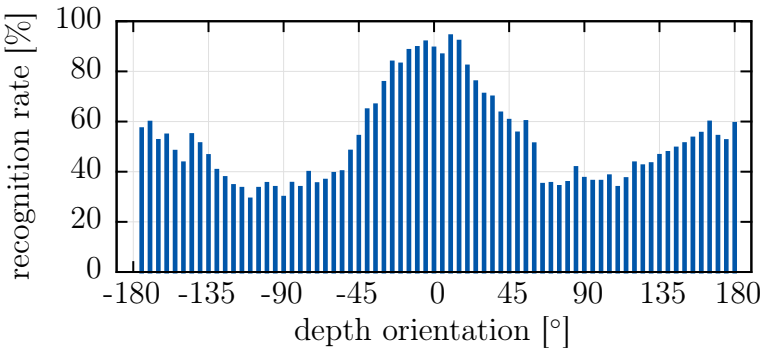
12.4.2 Effects of depth rotations

In addition to rotation in the image plane, mental rotation experiments also investigate rotations outside of it (for example, Shepard and Metzler, 1971). To achieve an analogous scenario, I train the space-edge-pattern-based object recognition system on the 0°-views of the first thirty objects of the COIL-100 database and let it recognize all available views of these objects. Because the objects were rotated on a turntable in front of the camera, this allows me to probe the behavior of the object recognition system specifically for depth-rotations.

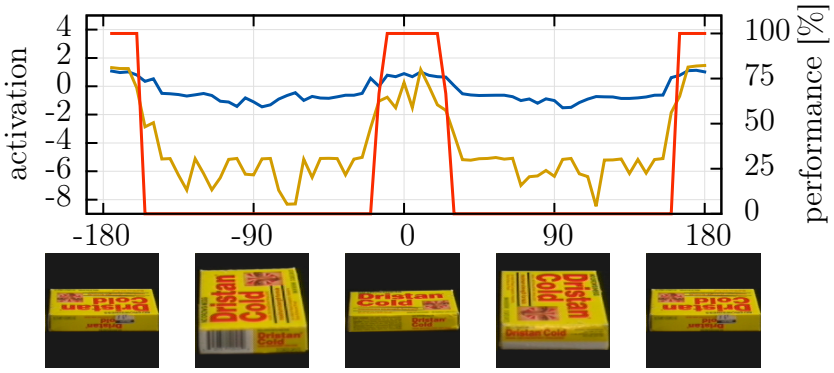
Figure 12.18(a) shows the performance averaged over all objects, plotted over the different depth rotations. Recognition rate shows a strong improvement close to the learned orientation (Pearson's χ^2 test for independence confirms that the result is significant: $\chi^2(71, N = 6085) = 802.2$, $p < 0.001$). This is most likely an effect of the appearance-based correlation function used for matching. Objects that vary with orientation correlate most strongly with the view in which they were learned. Slight rotations in depth only induce relatively small changes in the objects' appearances, and symmetric views have a similar effect. Figure 12.18(b) illustrates this for an object that only has limited symmetry (for rotations of approximately 180°). The activation of the label node corresponding to the object, an indicator for the match value, tends to be highest around the learned orientation and around the 180° image which has a highly similar appearance. Orientations that deviate more than approximately 30° from the learned and symmetric orientations induce a strong decline in activation.

Figure 12.18(c) demonstrates why the system has nonzero recognition rates for all depth orientations despite the appearance-based matching. Objects that are symmetric for all orientations induce high activation in their corresponding label nodes in any orientation because all views in the database are highly similar in appearance, and the object recognition system is therefore able to match them successfully.

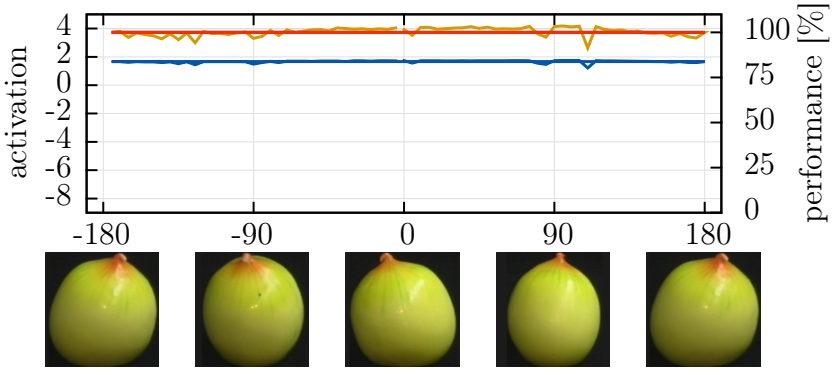
Besides to recognition rates, I also investigate the convergence times for the COIL trials. As before, I exclude trials in which the timeout was reached. Figure 12.19(a) shows response times for the trails, separately for correct and incorrect recognitions. Here it can be seen that correct recognitions cluster around approximately 50 seconds. This cluster is present over the entire



(a) Recognition rate for different orientations.



(b) Activation and performance for object 1.



(c) Activation and performance for object 2.

Figure 12.18: (a) shows the performance over depth rotation for all objects. (b) and (c) show for two exemplary objects the average activation at the end of the trial of the layer 1 (blue) and layer 2 (yellow) label nodes corresponding to the objects, as well as the recognition rate for the objects (red). Images of the objects at exemplary depth rotations are shown below the axis. Activation values have been averaged over all trials testing the shown objects.

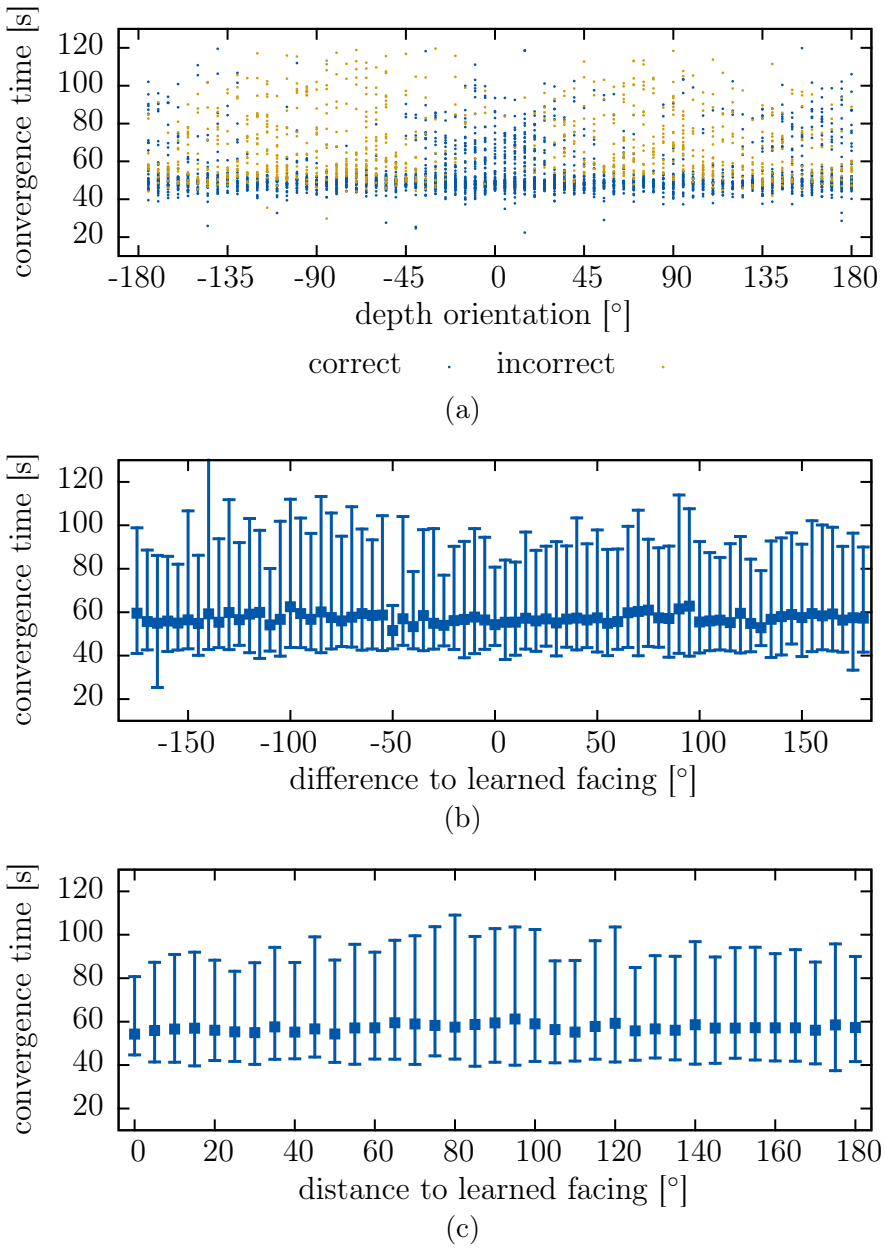


Figure 12.19: Convergence times over depth orientation. (a) shows the raw convergence times. (b) shows the mean response times for correct recognitions over the whole range of orientations. (c) shows the mean response times over the distance from the orientation in which the object was learned. In (b) and (c), squares indicate mean convergence times, while error bars indicate the 95% confidence interval of a gamma distribution fit onto the data for each orientation.

range of depth orientations. Figure 12.19(b) shows average convergence times for differences to the depth orientation of the object in the training view as well as error bars indicating the 95% confidence interval of a gamma distribution fitted on the points in each orientation bin. Figure 12.19(c) shows the analogous plot for the distance to the depth orientation of the object in the training view.

To determine whether there is a systematic relationship between depth orientation and convergence time I calculated Spearman's correlation coefficient, $\rho = 0.017$. This coefficient is very weak and not significant ($t(3496) = 0.980$, $p > 0.05$). For distance to the depth orientation of the learned stimulus, the correlation coefficient, $\rho = 0.048$, is still very weak. However, it is significant ($t(3496) = 2.854$, $p < 0.001$)

12.5 Discussion of the results

Here I characterize the object recognition system with respect to the human vision system. First, I discuss the results from the performance evaluation in Section 12.3. I then relate the data presented in Section 12.4 to results from the literature.

12.5.1 Performance

I have shown that the system is capable of recognizing objects in the absence of color information much better than the original histogram-based approach: in a comparable setup, recognition with space-edge patterns greatly outperforms the histogram-based system for the tabletop database, in particular in terms of recognition performance. Position estimation is also much more precise and even slightly exceeds the performance of the color histogram-based system (11.0 px vs. 13.5 px).

On average, the space-edge pattern system estimates the orientation of objects with lower accuracy than the histogram-based system. However, the histogram-based system estimates orientations in the smaller range of $[0, 180°)$, which also implies a smaller range of average orientation errors (errors range up to 90° vs. 180° for the space-edge pattern system). In addition, some of the common problems are masked by this restricted estimation range. As an example, take objects that are symmetric so that they look similar or even the same when rotated by 180°. This symmetry has no effect on the histogram-based object recognition system because both the 0° and 180° orientations are mapped to an orientation of zero degrees, which corresponds to an orientation error of 0° in both cases. By contrast, the

higher range of possible orientations in the space-edge patterns allows it to misestimate the orientation of such an object and thus incur an error of 180°. The clustering of orientation errors in the tests indicates that such cases are a large contributor to the overall increased orientation error.

The use of space-feature patterns allows the system to also estimate the scale of the input with respect to the learned object view. This was not possible in the histogram-based object recognition system, so that a comparison of the scale performance is not possible. However, on average, the scale error corresponds to deviations of approximately 10%, and the scale errors are grouped around zero. The distribution of deviations likely represents inaccuracies caused by the pose transformations, which effectively blur the patterns.

Recognition performance for the first thirty objects of the COIL-100 database is surprisingly high given that only a single training view is used. In part, this is due to objects that are symmetrical around the main axis of rotation, such as the objects 1 and 2, which have highly similar shape in some or all viewpoints (see Figure 12.18). To further increase performance, multiple object views may be learned. As described in Lomp et al. (submitted 2016), this may be achieved by assigning a label to each object view as usual and then summing the output of label nodes from all labels referring to the same object. However, my main goal for this particular evaluation was to provide evidence for the comparison to psychophysical results below, and this was therefore not implemented. Nonetheless, recognition seems to be stable for deviations of up to thirty degrees in either direction, which would imply that six object views should suffice to reliably recognize asymmetric objects.

The logistic regression model fitted onto the results from the transformed database indicates a weak but statistically significant relationship between transformations in the image plane and the recognition rate, even though all poses are treated equally in the pose-transformation and matching approach. In part, this effect may be explained simply by implementation constraints. At certain distances from the center, objects may no longer be fully inside the input region of the recognition architecture. This is true especially for larger objects. As a result, part of the object is not visible to the system, making recognition more difficult and thus reducing the recognition rate. At larger scales, similar effects may occur even for smaller objects, and objects closer to the center. At smaller scales, important details may be lost due to the spatial sampling, again making recognition more difficult. Together, these issues may explain the relatively large influence of scale.

For orientation, the dependence indicated by the logistic model was weak. To an extent, this is a result of the choice of model, because the relationship

does not appear to be monotonic even in the case where the distance to the upright orientation is used. On the other hand, the histogram of recognition rates indicates relatively large variation over orientation. This may be an effect of the coarse sampling of edge orientations, and implies that the representation may not be fully stable across all orientations.

Rotations in depth have a strong influence on recognition performance, as shown by experiments on the COIL database. The objects are well-recognized in the training orientation (more than 80% correct recognitions). However, the further objects are rotated, the further recognition performance drops. This is only the case for objects whose appearance varies with depth rotations.

For convergence times, I found some significant correlations with transformations in the image plane. Even in the cases in which I found significant effects, for example, for orientation, the correlation coefficient was very weak. These small effects are likely explained by the longer convergence times of incorrect recognitions (the mean convergence times of correct recognitions is 55.5 seconds, and is thus lower than the mean convergence times of incorrect recognitions, 62.2 seconds), and the increased error rates for some poses discussed above. Analogous to the results on recognition performance, the strongest effect on convergence time is therefore found for scale.

The effect of depth orientation on convergence times is very small (and significant only for the distance from the learned orientation). Because of the relationship between recognition rate and convergence time and the strong effect of depth orientation on recognition performance, a stronger relationship would be expected. In part, however, the relation may be weakened by symmetric objects that are recognized with comparable speeds at any orientations. By contrast, objects whose appearance varies strongly with orientation may lead to recognition trials exceeding the maximum duration more often because they cannot be matched to any known view. Because these are removed from the evaluation, this may further weaken the effect.

12.5.2 Relation to behavioral data

How do these results relate to biological vision systems? To an extent, this can be answered by looking at behavioral studies involving human subjects. Here, I separately review some relevant behavioral evidence on translation, orientation (both within and outside of the image plane), and scale. Note that this is not an exhaustive review of the available data, but rather a selection of relevant studies. A recommended (though somewhat dated) review can be found in Logothetis and Sheinberg (1996).

Effects of translation

Experiments provide evidence both for and against translational invariance. In the experiments by Kahn and Foster (1981) and Foster and Kahn (1985), participants fixated on a cross in the center of a screen. After fixation, they were presented with two random dot patterns at different distances from the fixation point. The two patterns were either the same, but possibly rotated and reflected, or were different altogether, which the participants had to indicate. Performance, both in terms of discrimination as well as reaction times, depended on the spatial separation of the stimuli.

Nazir and O'Regan (1990) performed experiments based on the study by Kahn and Foster (1981) and Foster and Kahn (1985). Stimuli were again random dot patterns, but participants first learned to recognize a specific target stimulus during a training phase. In this phase, the stimulus was presented at a fixed retinal position. Upon test, the target or distractor stimuli were presented at different retinal locations, and participants had to indicate whether the test stimulus was identical to the target. Error rates and response times increased when the position of the test stimulus differed from the position at which the target was learned. This again indicates a dependence of recognition performance on the stimulus position, which contrasts with my findings for the transformation and matching architecture. However, performance was above chance level even for positions in which the pattern had not been learned, indicating that matching is possible even at locations at which the target stimulus was never seen before.

Biederman and Cooper (1991) investigate recognition performance by measuring effects of translations on priming. In the experiments, subjects named line drawings of common objects. Participants fixated on a location on the screen, and objects were presented briefly so that no saccade to the target could be made. In the first block of trials (the priming block), images were placed at seemingly random positions. In the second block (the primed block), they appeared either in the primed position or a different one. Response times as well as naming errors were recorded for each trial. In the primed block, responses were generally faster than in the priming block. However, no significant effect of the stimulus position on response times or error rates was found, implying that recognition performance is independent of position. This largely matches the behavior of the object recognition system, although there is no long-term mechanism in the transformation and matching architecture that models priming effects, nor does the architecture perform recognition of such highly familiar objects.

Dill and Edelman (2001) asked participants to judge whether two images of animal shapes presented in sequence at different locations were the same or

different. The authors found some relatively weak effects of translation on the error rate that were most pronounced when the correct response was 'same'. Together with the conflicting evidence both for and against translational invariance from the literature, this leads the authors to conclude that the amount of invariance depends on the context of recognition.

Bowers et al. (2016) review literature on translation variance. The authors state that a dependence on translation is presumed in the literature. However, they also criticize the experimental design of many studies for translational dependence. They hypothesize that the exposure periods that are often used on the test trials are too short to allow for compensation of the translation. To address these concerns, the authors conduct experiments in which participants learned to recognize a small set of novel objects in fixed positions together with a label. An eye tracker ensured that participants only perceived the stimuli at the intended locations. Upon test, the objects were presented either at the learned position, or a different one, and subjects responded by producing the label for the object in their own time. The authors found that the correct naming rate in this scenario is far above chance level, and that the effects of stimulus position on the correct naming rate are small.

The study by Bowers et al. (2016) is perhaps the most fitting comparison to the transformation and matching architecture. Certainly, the assumption that a relatively long time is needed to compensate for translation fits with the relatively time-consuming recurrent process modeled by the transformation and matching architecture. The setting, learning to name a small set of novel stimuli also precisely fits the task addressed by the system. Furthermore, the stimuli used by Bowers et al. (2016), although structured, appear relatively nonsensical and thus unrelated to previously known stimuli, which means that influence from other known objects and general object knowledge is reduced. This fits the absence of contextual information from the transformation and matching architecture. In addition, the results on recognition performance qualitatively match the results for the transformation and matching architecture. Response times cannot be compared because they are not reported in the study by Bowers et al. (2016).

Effects of orientation

Effects of rotation on recognition performance have been studied in the context of mental rotation, which was first studied by Shepard and Metzler (1971). In this study, participants viewed pairs of three-dimensional figures made of cubes. Each pair showed either the same arrangement of cubes, rotated in depth or in the picture plane, or a different arrangement. The

participants were instructed to respond with 'same' if the objects were different views of the same object, and otherwise to respond with 'different'. Reaction times increased linearly with the angle of rotation between the two views. This linear relationship gives rise to the hypothesis that participants actively rotate the views in their mind analogous to the process of physically rotating the objects (hence the term *mental rotation*; however, this is by no means the only interpretation; see Thomas, 2016, for a discussion).

This result contrasts with my findings on the object recognition system. In the system, all orientations are initially equally active, and thus there is no systematic advantage for any specific orientation. The effects of orientation that I found, as discussed above, are likely artifacts of the implementation. At the very least, response times as well as recognition performance do not depend linearly on the amount of rotation and thus do not match the mental rotation results.

As with translation, the results may again depend on the task, and on the type and familiarity of the objects involved in it. Evidence for this is provided by Corballis et al. (1978), who asked participants to name images of a small set of letters and numbers. The letters were shown in different orientations, either in a normal or a mirror-reflected version. Significant effects on response time could only be found for the mirror-reflected stimuli. In the same paper, a second experiment is reported that uses the same stimuli. Participants pressed a button whenever the presented stimulus matched a target and pressed another button when it did not. Reaction times were again independent of the stimulus orientation. Error rates were low and not found to be related to stimulus orientation either.

When taking into account the implementation issues discussed above, the transformation and matching architecture shows behavior similar to these results. However, the naming task in the experiments by Corballis et al. (1978) uses objects (letters) with which the participants are highly familiar, and which are likely processed differently than other objects. Eley (1982) cited similar concerns about the stimuli used by Corballis et al. (1978), and thus replicated some analogous experiments with novel letter-like stimuli and confirmed that mental rotation was not employed and orientation had no significant effect on response times even for the novel stimuli.

Tarr and Pinker (1989) investigated these issues further using a small set of unfamiliar stimuli trained at a small number of orientations. These stimuli are chosen specifically to reduce the amount of diagnostic visual features that were present in the letter-like stimuli of Eley (1982) and similar experiments. Using this set, they asked participants to perform a variety of tasks in different experiments. In one experiment, the authors investigate naming performance and found a relationship between response time and orientation

that is consistent with mental rotation. This effect decreased as subjects became more familiar with the stimuli. The authors therefore conclude that objects are not represented by pose-invariant features, but rather are encoded in some view-dependent manner, much like the learned views of the object recognition system.

Bülthoff and Edelman (1992) also argue for a view-based representation. In their experiments, subjects were taught to recognize a novel stimulus, either a random wire object or a random 'amoeba' object, whose appearance depends strongly on the viewpoint. Training consisted of short motion sequences that showed the object rotating slightly around one of two training orientations. This was intended to allow participants to form a three-dimensional representation of the object. In the test set, the viewpoint of the object was varied either along the horizontal viewing plane (similar to the rotation of the COIL dataset used above), or along the vertical axis, and participants had to decide whether or not the test object was the same as the target object. Error rates increase strongly for horizontal deviations of more than 30° from the training view, providing evidence against three-dimensional representations and in favor of view-based ones.

Similar effects of depth may be found in the experiments I presented above. The overall recognition rate of the system strongly depends on the depth orientation and decays roughly around deviations of 30° to 60° around the learned orientation. Similarly, the average activation of the label nodes that correspond to objects whose appearance varies with rotation decays strongly when the orientation of the object varies by more than approximately 30°. In contrast to the study by Bülthoff and Edelman (1992), however, objects that vary with rotation are often no longer recognized correctly when rotated by more than 30°, and the error rates thus no longer match. However, participants in the experiments learned more training views than I used in my experiments, and it is possible that such additional views might allow for correct recognition in some cases.

Tarr (1995) describes an experiment that investigates the impact of depth rotation on response times in naming tasks. Novel objects were again constructed by arranging three-dimensional blocks to form stimuli similar to those used in the original mental rotation experiments. In various naming tasks, the author found that response times depended monotonically on the distance to the learned orientation. As described in the results section above, I did not find such behavior in the transformation and matching architecture, which has roughly constant response times for all depth orientations.

Effects of scale

Participants in the study presented by Larsen and Bundesen (1978) performed a same/different task on pairs of random polygon shapes presented in sequence. The shapes were scaled relative to each other, with one object being up to five times larger than other. The error rate was roughly constant over the different size ratios. However, response times increased linearly with the size ratio.

Jolicoeur (1987) presents a different task for which subjects studied a set of shapes in a training phase and indicated whether patterns are members of this training set in the test trials. Shapes were either arbitrary blobs or stick figures, and subjects learned patterns in a *small* or *large* version. Upon test, subjects were presented with shapes in both size categories. Response times as well as error rates in the test phase were lower for objects when they were presented in the size in which they have been learned, indicating again that recognition depends on scale. In these experiments, the effect also seems to be proportional to the scale ratio of the training and test stimuli.

As in the translation study, Biederman and Cooper (1992) investigated effects of size using a priming experiment. Participants again viewed simple line drawings of objects or animals and were asked to provide a name. In the first block of trials (priming block), the drawings were viewed at a either a small or large size. In the second block, the drawings were viewed in either the primed size or a different size. Participants fixated a cross at the center of the upcoming stimulus, and presentation of the stimulus was kept short to avoid saccades. Response times and error rates were again lower on the second block across all conditions, and there was no effect of scale, indicating that recognition is independent of size.

Most likely, the effect of size again depends on the task (Logothetis and Sheinberg, 1996). The effects of size in my experiments are more pronounced than the other effects, but are likely explained by implementation constraints as discussed above. The same argument I made for the independence of orientation may be applied here: no scale has an inherent advantage in the recognition process. This leads me to conclude that performance is independent of size, and the results I present therefore best fit the results presented by Biederman and Cooper (1992).

Chapter 13

Space-color patterns

In the transformation and matching architecture presented in the previous chapter, the estimation of pose solely serves to facilitate the recognition of the object in the image. At each stage of the architecture, the input is transformed by the current pose estimate, whereas the top-down estimate is transformed by the inverse of the current pose estimate. At the very end of the top-down path, the system provides a prediction of the input.

Thus far, this prediction serves no purpose beyond visual inspection of the estimation process. The goal in the present chapter is therefore to make use of this prediction to determine which parts of the input belong to the recognized object. This can be done by comparing the predicted feature value to the feature values at each location of the input image. If they are sufficiently similar, then the corresponding input location may be said to belong to the object.

With the space-edge patterns, such a comparison is problematic because the representation covers only a small part of the object (valid feature values are only present at locations of edges). Though these patterns may still serve to determine which locations of the input belong to the recognized object, they can only assign belongingness for object locations that have an inhomogeneous appearance, for example, locations at boundaries between differently colored object regions.

In the present chapter, I introduce a space-color representation that is used for transformation and matching together with the space-edge pattern architecture from the previous chapter. Each image location with a valid hue value that has sufficient brightness and saturation is entered into the space-color pattern. This means—at least for colored objects—that each point on their surface is part of the resulting representation and thus allows for a more holistic assignment of belongingness.

Since this belongingness is effectively a foreground-background segmen-

tation, though only for the object that was recognized, a natural approach is to use it to mask the input, much like the predicted shape in the shape channel of the histogram architecture is used to determine which parts of the input contribute to the color and edge histograms used for matching (see Chapter 3).

I begin the chapter with a formal description of the pattern extraction. Next, I describe the calculation of a belongingness function which formalizes the comparison between input and prediction. I then describe a quantitative evaluation of the on the tabletop dataset for different amounts of masking with the belongingness function. I show some qualitative demonstrations of the belongingness function, demonstrating recognition in highly occluded and cluttered scenarios and conclude the chapter with a discussion of the results, in which I also relate to the literature.

13.1 Pose-transformation and matching architecture

The architecture for applying pose transformations and matching for the space-color pattern is analogous to that described in Chapter 11. Because color is a feature uniquely determined for each location in the input image, multiscale extraction as described in the previous chapter is not necessary. This also implies that the feature extraction for each location is invariant to rotation, so that no transformations besides the spatial shift, rotation and scaling are necessary. Space-color patterns provide an additional channel for the space-edge pattern architecture. The individual contributions to the pose and identity estimation are combined by weighted addition, analogous to the different contributions of the feature channels in the histogram-based architecture (see Chapter 3). A sketch of the overall architecture is shown in Figure 13.1.

13.2 Extraction of space-color patterns

Space-color patterns are extracted from the input image as

$$P^{\text{scol}}(x,y,c,t) = \begin{cases} M(x,y,t) : & c = H(x,y,t) \wedge S(x,y,t) \geq \vartheta_S \\ 0 : & \text{otherwise,} \end{cases} \tag{13.1}$$

where c is a color, that is, a hue value from the HSV color space, $H(x,y,t)$ is the hue at image location x, y, $S(x,y,t)$ is the saturation at that location,

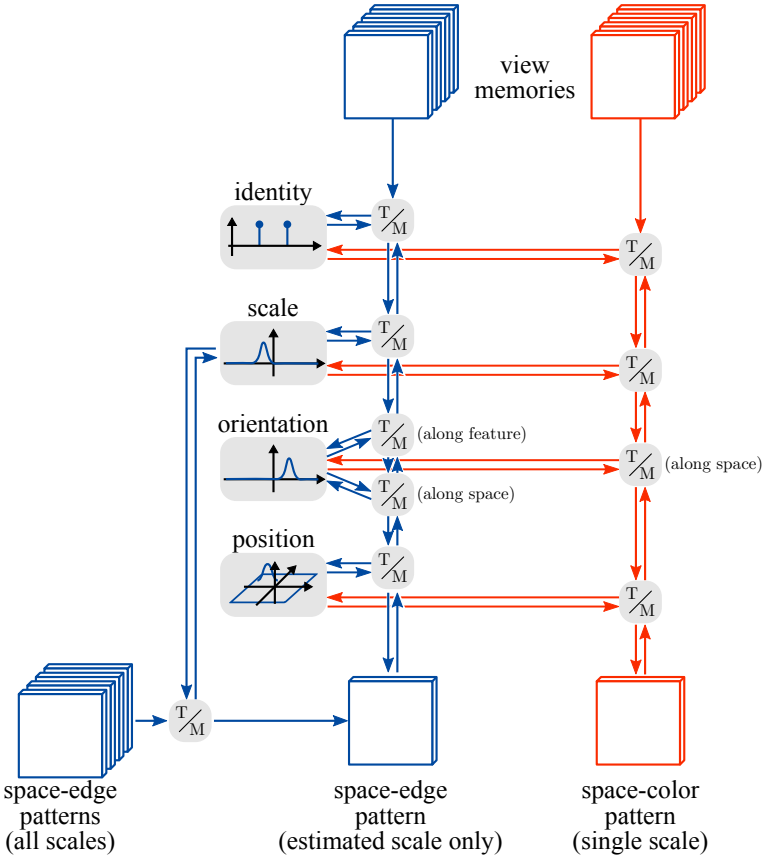
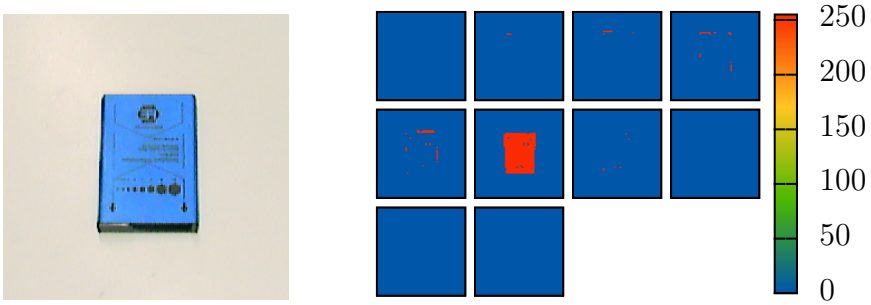


Figure 13.1: An overview of the object recognition architecture based on both space-edge patterns and space-color patterns. Boxes labeled "T/M" stand for transformation and matching operations.



(a) (b)

Figure 13.2: An example for a space-color pattern. (a) shows the input image, (b) the extracted space-color pattern. Here, each square stands for a different color range.

ϑ_S is a threshold for saturation, and $M(x, y, t) \in [0, 1]$ is a mask described in the next section. Normally, this pattern would also be convolved by a relatively narrow Gaussian; however, in the implementation, I sample color very coarsely by ten equidistant samples, and the pattern is further distributed in space by the transformations with the estimated pose. I thus forgo further convolution of the pattern. An exemplary space-color pattern is shown in Figure 13.2.

13.3 Belongingness and object boundaries

To determine belongingness, that is, which locations of the input match the predicted pattern and which parts mismatch the prediction, I correlate the color information from the transformed top-down pattern with the color in the input.

In the top-down pattern, all color entries are zero for locations that were part of the background during learning.[1] The function

$$P_{\text{mag}}(x, y, t) = \int_0^{2\pi} P_{\text{td,scol}}(x, y, c, t)\ dc \tag{13.2}$$

determines the magnitude of the top-down pattern, $P_{\text{td,scol}}(x, y, c, t)$, at each location, (x, y), by integrating over the range of possible colors, $c \in [0, 2\pi]$. A threshold, ϑ_{fg}, is applied to this magnitude using the Heaviside function $H(\cdot)$:

$$F(x, y, t) = H\left(\hat{P}_{\text{mag}}(x, y, t) - \vartheta_{\text{fg}}\right), \tag{13.3}$$

where the normalization of the magnitude function, indicated by the "hat", is the one defined in Chapter 3. Thus, F yields a value of one for locations considered to be foreground in the top-down pattern, and zero at other locations.

Foreground locations in the top-down pattern are matched to the input using the correlative method described in Chapter 3, that is, by point-wise multiplication of the functions. Applying a threshold, ϑ_{B}, to the result of this matching and filtering out background locations by multiplying with the foreground function, F, yields a belongingness function,

$$B(x, y, t) = F(x, y, t)\ H\left(\int_0^{2\pi} v_{\text{in}}(x, y, c, t)\ v_{\text{td}}(x, y, c, t)\ dc - \vartheta_{\text{B}}\right). \tag{13.4}$$

[1] This requires knowledge of what parts of the training image are background, which depends on the dataset. For the tabletop dataset, this is given because training images have a relatively uniformly white background which is filtered out by the hue and saturation threshold described in pattern extraction.

For this matching, patterns are normalized using an approach that differs from the one described in Chapter 3 because individual locations, rather than the whole pattern, are matched. The input, P^{scol}, is thus normalized to

$$v_{\text{in}}(x,y,c,t) = \frac{P^{\text{scol}}(x,y,c,t) - \frac{1}{2\pi}\int_0^{2\pi} P^{\text{scol}}(x,y,c',t)\ dc'}{\sqrt{\int_0^{2\pi} \left(P^{\text{scol}}(x,y,c'',t)\right)^2\ dc''}}. \tag{13.5}$$

Analogously, the transformed top-down pattern, $P^{\text{sh}}_{\text{td,col}}$, is normalized to

$$v_{\text{td}}(x,y,c,t) = \frac{P^{\text{sh}}_{\text{td,col}}(x,y,c,t) - \frac{1}{2\pi}\int_0^{2\pi} P^{\text{sh}}_{\text{td,col}}(x,y,c',t)\ dc'}{\sqrt{\int_0^{2\pi} \left(P^{\text{sh}}_{\text{td,col}}(x,y,c'',t)\right)^2\ dc''}}. \tag{13.6}$$

Values of one in B indicate input locations that are considered to belong to the recognized object. By contrast, $F(x,y,t)(1-B(x,y,t))$ yields ones at locations that are expected to be part of the object but have a feature value that deviates from the expectation. This can indicate, for example, perspective effects, but may also serve as an indicator of occlusion.

A natural use of the belongingness function is to apply a mask to the feature extraction (the $M(x,y,t)$ term in Equation 13.1). This mask may be calculated from the belongingness function as

$$M(x,y,t) = (1-m) + m\ \left[G_\sigma * B\right](x,y,t)\,, \tag{13.7}$$

where $m \in [0,1]$ is a factor that determines how much the mask attenuates the input.

Since the belongingness function specifies for each location in the input whether or not it is part of the object, the boundaries of the object may be defined as the borders between regions of different belongingness. In the implementation, I extract this information by first estimating the gradient of the belongingness function using a Sobel filter. The magnitude of the filter response is then blurred slightly by convolving with a Gaussian, and the result is thresholded to obtain locations that are considered to be part of the boundaries of the object.

13.4 Evaluation

To measure how masking the input with the belongingness function impacts on performance, I trained the architecture at $\frac{1}{4}$ spatial resolution on the table-top database with the training protocol described in Section 11.4. During training, top-down masking was disabled by setting $m = 0$ in Equation 13.7

(the masking term that separates background and foreground by applying a threshold to the saturation of the image was still active). The procedure for evaluating performance follows the one described in Section 11.4. Foreground and belongingness functions as well as a mask analogous to that of the color channel were also calculated for the edge channel presented in Chapter 12.

13.4.1 Tabletop performance

Figure 13.3 shows the performance obtained on the tabletop database for different masking factors (m in Equation 13.7). This figure shows that as the mask factor increases, performance decreases for most measures. The scale error deviates from this slightly, showing a small improvement for mask factor 0.25 over no masking. Rotation deviates more strongly and does not show a clear trend. Overall, however, top-down masking seems to worsen the performance of the architecture, in particular for mask factors of 0.5 and above.

It is, however, worth noting that the system outperforms the histogram-based object recognition system presented in Chapter 3 at the same feature resolution when no masking is applied. The space-color and edge orientation system achieves a recognition rate of up to 73.1% with ten color samples and eight edge orientations samples. Comparable results, that is, a recognition rate of 73.0% were achieved with the histogram architecture only when 36 color samples and 18 edge orientations were used (Lomp et al., submitted 2016). At the closest sampling rate investigated by Lomp et al. (submitted 2016), that is, 18 color samples and 9 edge orientation samples, the histogram architecture achieved a recognition rate of 61.5%.

13.4.2 Demonstration of masking

The images in the tabletop database do not contain any distractors or occlusions, nor is the background complex. A quantitative evaluation of the belongingness function is therefore, unfortunately, outside the scope of the present work. Instead, I qualitatively demonstrate the belongingness function in this section. The demonstrations are based on the same system I used to evaluate performance on the tabletop database for different mask factors. Unless otherwise stated, I use a mask factor of zero ($m = 0$) because this yielded the highest recognition rates in the experiments above.

Figure 13.4 shows an example of the belongingness function and outline after the system is converged. The input image, Figure 13.4(a), is a test image from the tabletop database with artificially generated occlusions and distractors. The added colors are sampled from objects in the tabletop

database. The resulting belongingness function is shown in Figure 13.4(b) (red indicates locations for which the function is one, blue indicates locations for which it is zero). In Figure 13.4(c), the belongingness function is used to mask all locations of the image that have a belongingness of zero (black indicates such regions). Note that in the implementation, the belongingness is only calculated at $\frac{1}{4}$ of the resolution of the image. The mask was therefore interpolated to achieve the same resolution as the input. Despite visible artifacts from this interpolation process, the belongingness matches the object well. Figure 13.4(d) shows the outline of the object, that is, the borders between belongingness values calculated as described above. Figure 13.4(e) shows the outline in yellow, superposed onto a darkened version of the input image (the region with belongingness values of one is shown more brightly). Again, interpolation artifacts are visible in the outline, but it appears to otherwise match the object well.

Figure 13.5 illustrates the convergence process of the mask for the same image. The visualizations are analogous to Figure 13.4. Initially, the pose estimates are unspecific (top row), and thus all locations of the image that exceed the saturation and value threshold have a belongingness of one. When pose and label estimates become more specific (middle panel), some locations are masked because they lie outside the range of the more specific top-down prediction. When pose is closer to convergence (bottom panel), the system is focused mostly on the object. However, the blue distractor blobs make the estimate of rotation ambiguous. As a result, the rotation estimate is still unspecific and leads to a top-down prediction that contains the top-down pattern in all orientations. Over time, other pose estimates become more accurate, decreasing the ambiguity of the object's orientation. Finally, the architecture converges to the state shown in Figure 13.4.

Figure 13.6 shows more examples of the belongingness function for different distractor and occlusion setups. In Figure 13.6(a), the system has focused on the green part of the input image. The system has recognized this region as the *deodorant*, an elongated, green object, likely because the low spatial and feature resolution is not sufficient to make a proper distinction. Apart from the issue of resolution, this example illustrates another problem: if we accept that the green section may be recognized as the *deodorant*, then both answers (*deodorant* and *bit box*) would be valid, and the decision of which object should be recognized must be provided externally.

Figure 13.6(b) shows another example of a successful recognition. The image has been masked with random color noise. Some of the noisy locations are also recognized as part of the object because their color is close enough to that of the perceived object, and some of these lie outside the boundaries of the object because the object is tilted backwards slightly, and the architecture

does not estimate such a perspective transformation. As a result, the top-down prediction is larger than the object in the input image.

Figure 13.6(c) shows the mask obtained for a simulated complex background made of colors sampled, again, from the training images in the tabletop database. Here, the system again has difficulty determining the proper object pose due to the distractors close to the object. Though the mask is relatively fitting, the system did not converge to a final estimate. The pose estimate at convergence is still relatively broad, and the estimated rotation is off by approximately 90°, leading to the inclusion of part of the background. Nonetheless, the label was identified correctly. However, the system is susceptible to relatively small changes in the composition of this complex background (see below). If the background colors are more similar to objects in the database, recognition of the label is no longer achieved and, instead, a background location is recognized. This may again be caused at least in part by the low spatial and feature resolution at which the input is sampled.

Figure 13.7 shows a slight variation of the complex background. On its own, the system fails to recognize the object and instead focuses on part of the background. However, if the label is specified by an external boost to the correct label node, the object is located properly. In Figure 13.7(b), the belongingness obtained with a masking factor of $m = 0$ is shown. Visually, the mask fits the input well, even though in this case the pose estimate at convergence was still broad and not all layer two fields were above threshold. When a masking factor of $m = 0.25$ is applied, the belongingness shown in Figure 13.7(c) is obtained. The result is similar to the unmasked case, however, in this case, suprathreshold peaks form in the second layers of the pose estimation, making the belongingness more accurate.

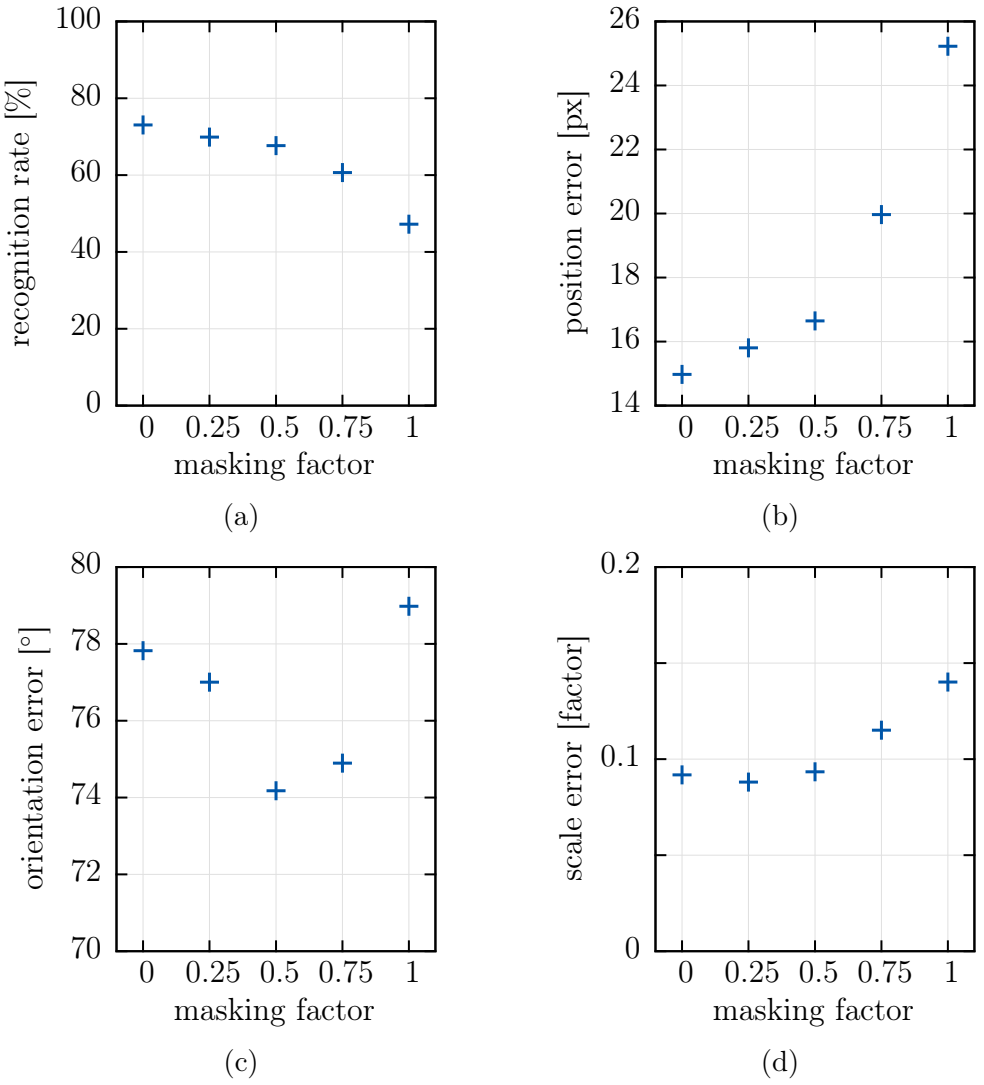
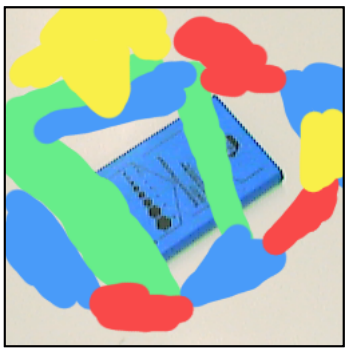
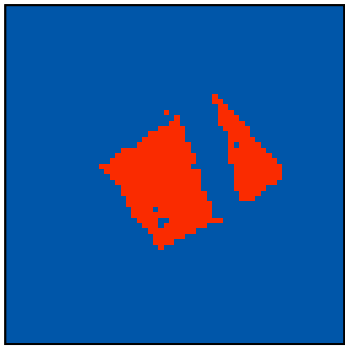


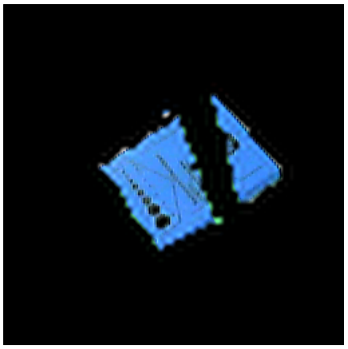
Figure 13.3: The figure shows the average recognition rate (a), average position error (b), average orientation error (c) and average scale error (d) for different masking factors.



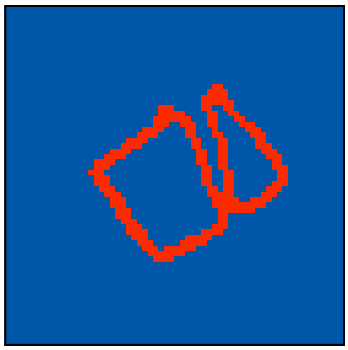
(a) Input image.



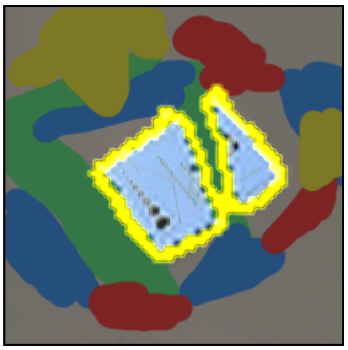
(b) Belongingness.



(c) Masked input.



(d) Object outline.



(e) Visualization of belongingness and outline.

Figure 13.4: Belongingness and outline for a test input.

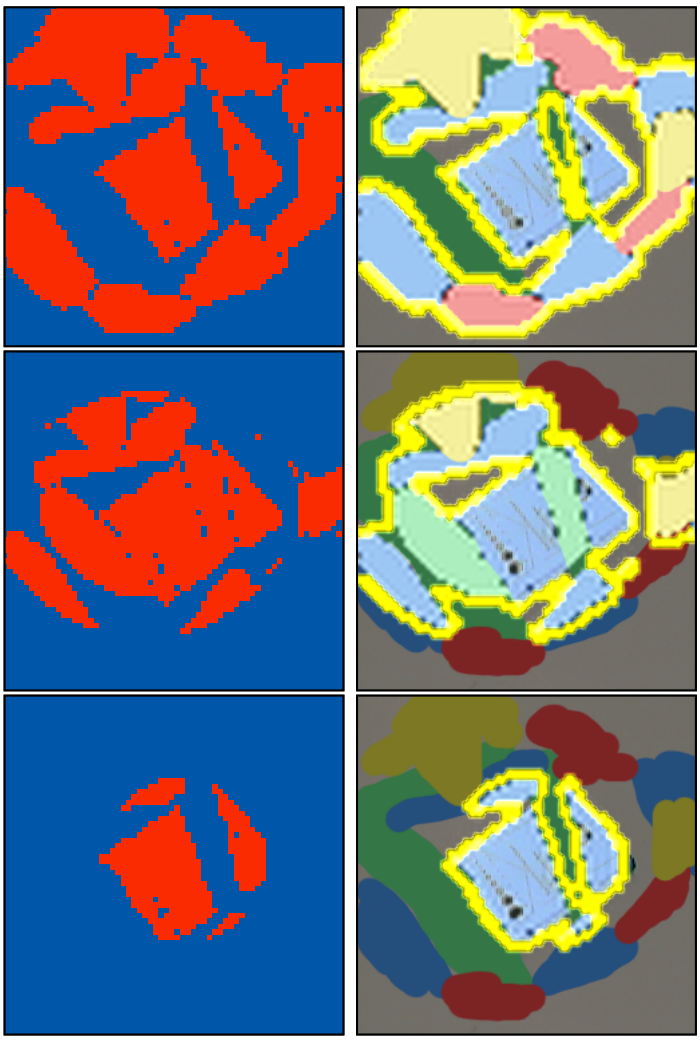


Figure 13.5: This figure shows the evolution of the belongingness over the recognition process (left column) and a corresponding visualization (right column). Time increases from top to bottom. The final state is shown in Figure 13.4.

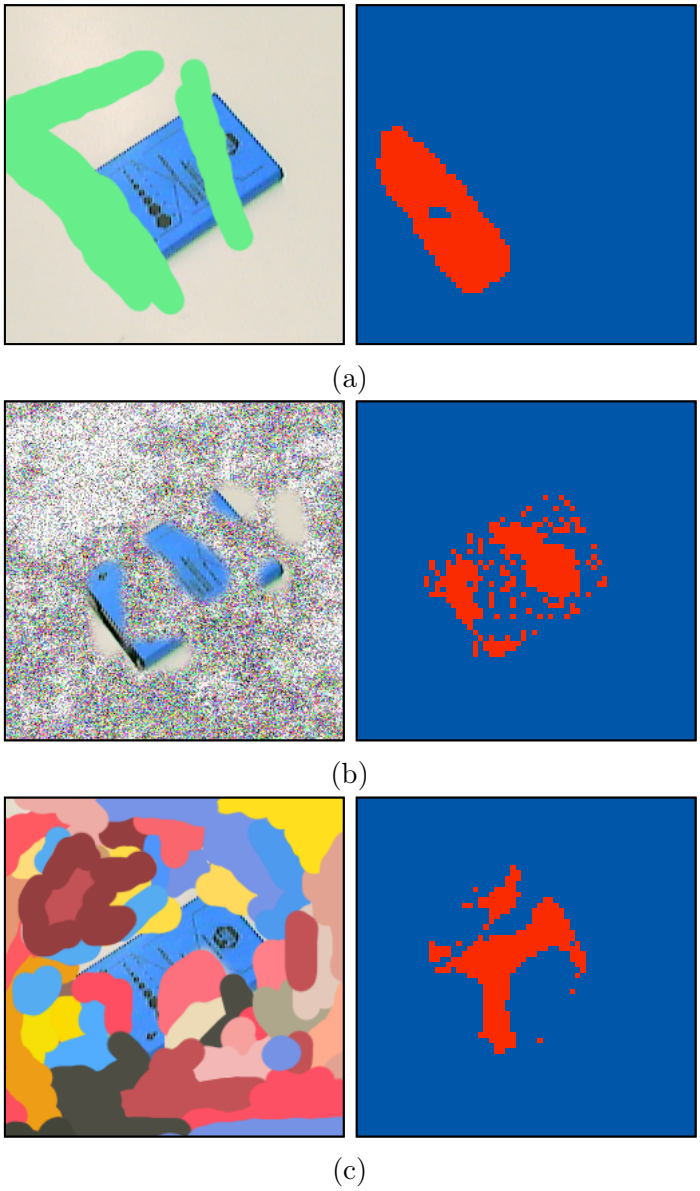
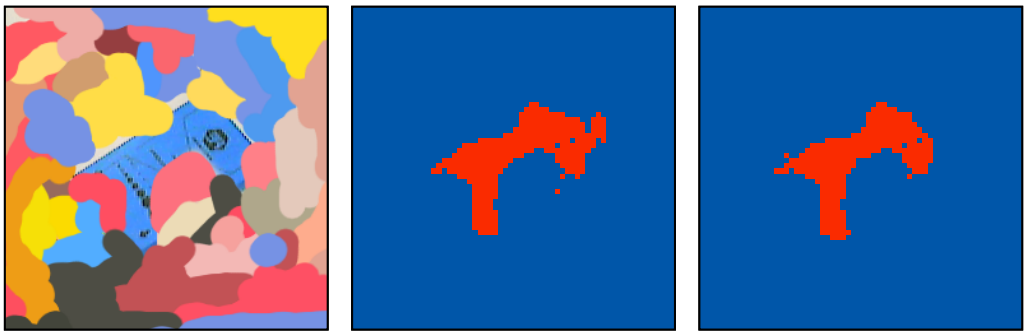


Figure 13.6: This figure shows further examples of the belongingness function for different inputs. The left column shows the inputs themselves, the right column shows the belongingness, where red areas again stand for belongingness values of one, and blue for belongingness values of zero.



(a) (b) (c)

Figure 13.7: Belongingness ((b) and (c)) for a complex background (a) that is based on minor modifications of Figure 13.6(c). The correct label was provided to the system.

Chapter 14

Discussion of space-feature patterns for object recognition

In Part III, I have shown that pose-transformation and matching of space-feature patterns is possible by extending the principles used for matching localized feature histograms (see Chapter 3). I have also shown that this extension leads to higher recognition rates as well as several other beneficial aspects. One such aspect is the preservation of the spatial arrangement of feature values. The histogram-based representation discards this arrangement, making it susceptible to scrambling, whereas space-feature patterns preserve it, allowing for a more precise estimation of the transformation parameters. In addition, this enables the estimation of orientation on the full 360° range (as opposed to the 180° range available with localized histograms), and the estimation of scale, which was not available in the histogram-based architecture.

Another aspect is that space-feature patterns more closely resemble the space-feature activation patterns common to dynamic field theory. By contrast, the localized histograms require representing the frequency of feature values with an individual neuron, which is not part of the mathematical framework of dynamic field theory. However, the concrete patterns I use are also graded representations and thus deviate from the space code principles underlying dynamic field theory, but this mainly serves to address issues arising from the coarse sampling of the feature spaces. Regardless, the resulting transformation and matching system can be used to match space-feature representations such as the output of a space-color field.

As I have discussed throughout the previous two chapters, the localized edge orientation histograms system mainly support the estimation of orientation, but contribute very little to the estimation of identity. This leaves color as the main diagnostic feature, making recognition of non-colored objects (or

grayscale images) difficult or even impossible for the object recognition system. Space-edge patterns, on the other hand, recognize objects with much greater success because the spatial arrangement of the edge orientations is preserved. This is a step towards reproducing the visual system's ability to recognize objects based on line drawings (Biederman and Ju, 1988), and makes the system independent of color. Although this constitutes an improvement over the histogram-base system, recognition rates are low when compared to human performance. Likely, the human vision system employs additional more complex features that aid and improve recognition (Rossion and Pourtois, 2004). One of these features is color (Rossion and Pourtois, 2004), which I have also shown to improve the recognition performance of the system presented here. This improvement is relatively small, but this is likely due to the coarse sampling of the feature space chosen due to computational constraints.

Though the space-edge patterns are based on the image gradient, this is only meant as a shorthand for the responses of simple and complex cells found in the visual cortex. These cells react maximally to bars of specific orientations at specific retinal locations, a behavior that is mirrored by each location in the space-edge patterns. I therefore also consider them more neurally plausible than the edge orientation histograms. This has allowed me to provided a more rigorous characterization of the system, and to draw conclusions about the relationship between the transformation and matching architecture and human behavior. In Chapter 12, I have demonstrated that performance of the system in terms of recognition rate and convergence times is largely independent of the aspects of pose estimated by the system, whereas pose aspects not estimated by the system, such as depth rotation, systematically impact recognition performance. The literature provides evidence for and against pose dependent performance, and which outcome is observed largely depends on the task being performed by participants. In experiments probing tasks most similar to the task solved by the object recognition architecture, performance is often pose dependent, though this is not the case for all studies. Overall, this evidence implies that the connection to the vision systems in the brain is questionable.

The results I present in Chapter 12 have an interesting implication for the debate on how objects are represented in the brain. One common interpretation of pose-invariant recognition performance in humans is that the visual system extracts diagnostic features from the input in such a way that the description itself is invariant to changes in the objects' poses (see, for example, Eley, 1982; and, Peissig and Tarr, 2007). When recognizing an object, the extracted features are compared with feature descriptions stored in memory, and the object is recognized with little to no impact on recognition

performance because the feature is mostly invariant to pose changes.

On the other side of the argument, pose-dependent object recognition performance in humans is often interpreted to mean that the vision system represents objects in a view-dependent way (see, for example, Bülthoff and Edelman, 1992; and, Peissig and Tarr, 2007). View-dependent performance arises when the perceived stimulus does not fall exactly onto one of the learned views, either because of reduced match values (Bülthoff and Edelman, 1992), or as a result of a processes similar to mental rotation (Ullman, 1989; Graf, 2006) that aligns the learned object representation with the input.[1]

The results I show offer a third possibility that is, to my knowledge, not considered in the literature. The view-based pose-transformation and matching system achieves alignment without sequential transformations like mental rotation. Instead, it uses all possible pose transformations in parallel, aligning the input and the top-down prediction at the same time using the superposition idea presented by Arathorn (2002). As my experiments confirm, this results in pose-independent performance both in terms of recognition rates and convergence times, thus contradicting the common interpretation that view-based systems must show pose independent performance while the performance of alignment based systems must be pose-dependent.

A related argument in the literature is that the qualitative differences in recognition performance for different tasks speak either to the presence of multiple recognition systems in the brain (Logothetis and Sheinberg, 1996 provide a good, though again somewhat dated account of relevant findings of evidence for multiple recognition systems). Other authors that argue for a continuum of behaviors that depend on the task include Bülthoff et al. (1995) and Dill and Edelman (2001). More recent findings suggest the presence of two systems, one system for holistic processing, wherein objects are stored in a view-dependent manner, and another system which encodes objects by their components (Peissig and Tarr, 2007).

Together, this can be interpreted to mean that the object recognition system presented here is a part of a larger object recognition system. Fast feedforward processes akin to the HMAX architecture (Riesenhuber and Poggio, 1999) may provide additional recognition capabilities, for example, by providing input to the identity representation. Such input can improve identification and thus also lead to more precise localization over time. Such a combined system might lead to improved performance near the learned view due to the supporting input to the identity representation. It might also ex-

[1]Note, however, that any alignment processes used in object recognition are likely different from mental rotation, as several studies show that mental rotation and object recognition are solved by different systems in the brain (Gauthier et al., 2002; Peissig and Tarr, 2007).

plain the brain's ability to identify unfamiliar stimuli in novel poses as well as its pose dependent performance in some tasks, making the combined system more compatible with experimental evidence.

Prior work in the field of computer vision has also used edge information for object recognition. The most related approach may be the original scale invariant feature transform (SIFT) approach (Lowe, 1999, 2004). In this approach, oriented keypoints are extracted, and histograms of gradient orientations for each keypoint describe the region around it, somewhat analogous to the space-edge patterns. However, the structure used for SIFT is not as rigid—descriptors are associated with keypoints rather than space—and the matching process is therefore also solved differently. Keypoints are matched to the most similar ones in the stored object images, and a generalized Hough transformation (Ballard, 1981) finds object hypotheses from these matches. The transformation between the image and the stored image is calculated by solving the overdetermined equation system resulting from the transformation equations between the different points of sufficiently matching hypotheses.

In terms of recognition performance, deep convolutional networks (LeCun et al., 1998; Hinton et al., 2006; Bengio et al., 2007) present the state of the art at the time of writing. They have achieved performance equal to and even exceeding that of humans (Cireşan et al., 2012), though their relation to biological networks is unclear (Schmidhuber, 2014). Pose is usually discarded in such networks. However, Jaderberg et al. (2015) show that extending a deep convolutional network by a *spatial transformer network* that learns to transform inputs to minimize the impact of pose may improve performance. This bears some similarity to the transformation and matching architecture, as both networks transform the input into a spatial arrangement that eases the matching process. While the similarities are somewhat abstract, this demonstrates the usefulness of pose estimation even for feedforward processes. As the authors themselves argue, an alignment of reference frames, possibly by a recurrent process, may prove useful for many systems. I discuss the relationship between such convolutional networks and the systems I present further in Chapter 15.

I have also demonstrated that the top-down prediction made by the pose-transformation and matching architecture may deliver precise information on which regions in the input image correspond to the recognized object. In cases in which the recognition process is successful, visual inspection suggests that this information is highly accurate. However, using this information to exclude non-object locations from the matching process through masking leads to reduced performance. This is due to problems that arise when the representation of one object is similar to part of another object. In

this case, if the smaller object happens to temporarily gain an advantage during the recognition process, all evidence against the false recognition it is excluded from matching because it is considered to be non-object. This makes recognition of the larger, containing object unstable and thus leads to worse recognition performance.

A similar issue has been reported by Borenstein and Ullman (2002) for a top-down segmentation system. The authors use pre-segmented templates of parts of a class—horses—and match them to a new input image. They find that excluding the background from the matching process reduces susceptibility to noise, but also decreases accuracy in the placement of the individual parts. The authors solve this by including the border between figure and ground in the matching equation, an approach that could be included in the object recognition system I present here in future work.

This problem was not observed in the histogram-based transformation and matching approach for two reasons. First, the mask from the shape channel is relatively coarse due to low spatial sampling rates as well as repeated blurring by Gaussians as well as the transformations by the pose estimate. Second, color histograms are sufficiently discriminative, so that the ambiguities between the learned objects are reduced and cases in which one object is found inside another occur only infrequently, if at all.

Part IV

Conclusion

Chapter 15

Major contributions and outlook

The histogram-based object recognition system described in Faubel and Schöner (2009, 2010) and Lomp et al. (submitted 2016) constitutes the first neural-dynamic process model for Arathorn's map-seeking circuit (Arathorn, 2002) and has demonstrated the feasibility of such a model. Nevertheless, some parts of the model required further investigation because they were not fully realized with neural principles. In addition, the architecture's link to biological vision was not established experimentally. My main contribution is addressing these issues. To that end, I present two extensions of the histogram-based architecture in Parts II and III. I have already discussed in detail how these extensions relate to the literature in their respective parts (Chapter 10 and Chapter 14); here, I relate the two parts of my thesis to each other and discuss how they contribute to a more neurally grounded picture of the map-seeking circuit and its implementations in dynamic field theory.

In my first major contribution, I replaced nonneural control mechanisms in the original implementation of the object recognition architecture by a neural-dynamic behavioral organization system based on dynamic field theory. This allows the architecture to recognize multiple objects in succession by detecting when a new input is presented to the system, and by detecting when the recognition of a new input is complete. In contrast to the original implementation, these detections happen fully within the neural dynamics of the system. This modification therefore increases the degree to which the architecture can claim to model the processes of biological vision.

My second major contribution is the integration of this modified object recognition model with a biologically plausible model for scene representation. This integration addresses another nonneural part of the previous implementation of the object recognition system, which required the input region used for recognition to be provided externally. In the combined system I present, this region is selected by the attentional control of the scene

representation model. The integration works in both directions, and the recognized identity for the attended region is communicated back to the scene representation, which forms a working memory representation of this recognition. Overall, this provides a large-scale architecture for object recognition that includes cognitive capabilities absent from many other models. Furthermore, this model is biologically plausible and entirely built within the framework of dynamic field theory. My demonstration of the feasibility of such a large-scale integration also provides support for one of the central claims of dynamic field theory, namely, that its principles scale to larger architectures, and that the principles are modular so that individual architectures can be combined without loss of functionality (Lomp et al., accepted 2016).

My third major contribution is the introduction of space-feature patterns for representing object views. This representation is closer to the space-feature patterns commonly used in dynamic field theory than the representations used by the original histogram-based object recognition approach. It also shares similarities with the simple cells found in visual cortex (Hubel and Wiesel, 1962, 1968). Together, this again brings the pose-transformation and matching architecture closer to neural plausibility. I have demonstrated that such space-feature patterns are feasible for recognition, and that they lead to better performance than the histogram-based approach when color is not available for matching. Further, I have shown that space-feature patterns enable scale estimation, which was not part of the histogram-based object recognition system.

The implementation of the pose-transformation and matching architecture for space-feature patterns uncovered some issues with the original neural-dynamic implementation of the map-seeking circuit. One important issue is the tendency of the matching process to run into local maxima. I have addressed this by adding an inhibition scheme that increases the competition between different pose and identity candidates over time during the process of recognizing an object. This scheme resembles the competition function of Arathorn (2002) more closely than the fixed inhibition scheme of the two layer architecture in the original neural-dynamic implementation of the map-seeking circuit.

Previous evaluations of the object recognition architecture performed by Faubel and Schöner (2009, 2010) and Lomp et al. (submitted 2016) focused on the architecture's performance in a robotic context rather than an evaluation of the architecture's relation to biological vision. As discussed above, this was justified because the feasibility of the approach first needed to be demonstrated before establishing such a relation.

While I still evaluate the architecture in a robotic context, I have provided

a more thorough characterization of how pose and other factors influence the architecture's performance. Together with the increased biological plausibility obtained by replacing the representation used by the system, this allows me to compare data from the architecture with behavioral experiments. I could therefore rule out that the architecture's behavior fits the mental rotation picture, and provide evidence that this view-based system achieves pose-invariant performance despite using alignment processes, a possibility largely discounted in the literature. An important point to note is that even though I focus on the neural-dynamic implementations of the transformation and matching approach, these results may also be extended to the approach proposed by Arathorn (2002). While Arathorn did not provide a process model which could be tested in the same way, the finding that performance is independent of pose likely transfers to his approach because the underlying principle of weighted superpositions is common to the different implementations.

I have also shown how color may be reintegrated into the architecture, and that it increases the discriminative power of the system. I demonstrated how space-color patterns may be used to match the top-down prediction generated by the object recognition system to the input, and how the result of this match can be used to distinguish image locations that are part of the object from non-object locations. Such an assignment was made possible because, in contrast to feature histograms, space-feature patterns preserve the spatial arrangement of the encoded feature values, and because they may encode the absence of such information. I have also shown that removing non-object locations from the matching process impacts performance negatively which calls into question the role of top-down predictions in segmentation for object recognition.

Overall, I have developed an integrative system for object recognition based on biological principles. This is by no means a complete model of human perception; there are more processes at work in the brain, for example those that implement Gestalt laws. Neural dynamic models for such implementations exist (for example, Grossberg and Mingolla, 1985), and an integration of such models might provide further insight.

One practical issue with the systems I present in my thesis is their high computational demand. A large factor in this are the calculations involved in approximating the dynamics of fields, in particular the convolutions required for calculating interaction in fields with three or more dimensions. This strongly affects the combined scene representation and object recognition architecture, which contains several three-dimensional fields. However, the convolutions and cross-correlations in the transformation and matching architecture also cause long cycle times. This, in turn, implies large (and

thus slow) timescales for the neural dynamics, and the systems are no longer capable of operating in real time scenarios. However, this problem is specific to the discrete implementation on a CPU that is, in essence, a sequential system. Although modern processors support multiple parallel threads of execution and my implementation makes use of this, such a parallelization does not nearly match the breadth of parallel processing in the brain. The most time-consuming operations of the architectures—the convolutions and correlations—can be computed entirely in parallel by a network of neurons, so that an implementation of the object recognition system in neural hardware may achieve much faster speeds.

Deep networks, the current state of the art in object recognition classify inputs using purely feed-forward connectivity. This gives them an advantage in terms of processing speeds and at the same time achieves classification rates that exceed the capacities of the systems I presented here. However, the feed-forward nature of these networks also implies that there is a deterministic relationship between the image and the network's output. Cognition is not part of these networks because it would require the networks to have an internal state that also influences the output of the system. A question is, therefore, whether the cognitive aspects of the systems I present here could also be provided for such feed-forward systems.

For the integration with the scene representation architecture, this would certainly be possible. A convolutional network might simply take the place of the neural-dynamic object recognition system, and as presented in Part II, the saliency mechanisms of the scene representation architecture may provide input regions which are then recognized by the convolutional network. As a result, the combined architecture may label objects in natural scenes without exhaustively applying the still costly recognition machinery. One of the main questions in such a fusion would be whether the recognition result of the network can be seen as instantaneously available, as is the case for the color extraction in the color channel of the scene representation architecture, or whether a process model for the convolutional network is appropriate or necessary.

Another aspect concerns the integration of the neuro-dynamic object recognition system with a deep convolutional network. The learned filters on the first layer of the convolutional network are certainly candidates for providing more discriminative features for object recognition and pose estimation. Their responses also constitute a space-feature pattern that could be input into the transformation and matching architecture described in Part III. However, the metric of the feature space is unclear, and it is therefore also generally unclear how the different transformations affect this representation. For position, responses may be spatially shifted as described in Part III be-

cause the same filters are used throughout the first layer of a convolutional network. How rotation and scaling may be applied, however, is unclear. The simple shift along the feature dimension used to rotate the edge orientations, for example, does not have an obvious analog for such filters. Instead, the effect of rotations on the representation would likely have to be learned. How this could be done is unclear at present. If achieved, however, pose may conceivably be used to align the input to improve the match achieved by the convolutional architecture. As discussed before, such alignment has been shown to be beneficial for recognition performance (Jaderberg et al., 2015).

Chapter 16

Bibliography

S.-i. Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, 27:77–87, 1977.

D. W. Arathorn. Recognition under transformation using superposition ordering property. *Electronics Letters*, 37(3):164–166, 2001. ISSN 00135194. doi: 10.1049/el:20010123.

D. W. Arathorn. *Map-seeking Circuits in Visual Cognition: A Computational Mechanism for Biological and Machine Vision*. Stanford University Press, 2002. ISBN 9780804742771.

D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems*, 19(1), 2007.

M. Berger, C. Faubel, J. Norman, H. Hock, and G. Schöner. The Counter-Change Model of Motion Perception: An Account Based on Dynamic Field Theory. In A. E. Villa, W. Duch, P. Érdi, F. Masulli, and G. Palm, editors, *Artificial Neural Networks and Machine Learning - ICANN 2012*, pages 579–586. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33268-5.

E. Bicho, P. Mallet, and G. Schöner. Target representation on an autonomous vehicle with low-level sensors. *The International Journal of Robotics Research*, 19(5):424–447, 2000.

E. Bicho, L. Louro, N. Hipólito, and W. Erlhagen. A dynamic field approach to goal inference and error monitoring for human-robot interaction. In

K. Dautenhahn, editor, *Proceedings of the 2009 International Symposium on New Frontiers in Human Robot Interaction*, pages 31–37, Edinburgh: AISB 2009 Convention, Heriot-Watt University, 2009. ISBN 1902956850.

I. Biederman. Recognition by components: A theory of human image understanding. *Psychological Review*, 94(2):115–117, 1987. ISSN 0033-295X. doi: 10.1037/0033-295X.94.2.115.

I. Biederman and E. E. Cooper. Evidence for complete translational and reflectional invariance in visual object priming. *Perception*, 20:585–593, 1991. ISSN 03010066. doi: 10.1068/p200585.

I. Biederman and E. E. Cooper. Size invariance in visual object priming. *Journal of Experimental Psychology: Human Perception and Performance*, 18(1):121–133, 1992. ISSN 0096-1523. doi: 10.1037/0096-1523.18.1.121.

I. Biederman and G. Ju. Surface versus edge-based determinants of visual recognition. *Cognitive psychology*, 20(1):38–64, 1988. ISSN 00100285. doi: 10.1016/0010-0285(88)90024-2.

E. Borenstein and S. Ullman. Class-specific, top-down segmentation. *Computer Vision—ECCV 2002*, 2351:109–122, 2002. ISSN 16113349. doi: 10.1007/3-540-47967-8.

A. Borji and L. Itti. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.89.

J. S. Bowers, I. I. Vankov, and C. J. Ludwig. The visual system supports online translation invariance for object identification. *Psychonomic Bulletin & Review*, 23:432–438, 2016. ISSN 1531-5320. doi: 10.3758/s13423-015-0916-2.

H. H. Bülthoff and S. Y. Edelman. Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proceedings of the National Academy of Sciences of the United States of America*, 89 (1):60–64, 1992. ISSN 0027-8424. doi: 10.1073/pnas.89.1.60.

H. H. Bülthoff, S. Y. Edelman, and M. J. Tarr. How Are Three-Dimensional Objects Represented in the Brain? *Cerebral Cortex*, 5(3):247–260, 1995. ISSN 1047-3211. doi: 10.1093/cercor/5.3.247.

G. Carpenter and S. Grossberg. Adaptive resonance theory. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, number 617, pages 1–12. MIT Press, Cambridge, Massachusetts, 2nd edition, 2003.

D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.

M. C. Corballis, N. J. Zbrodoff, L. I. Shetzer, and P. B. Butler. Decisions about identity and orientation of rotated letters and digits. *Memory & cognition*, 6(2):98–107, 1978. ISSN 0090-502X. doi: 10.3758/BF03197434.

J. J. DiCarlo, D. Zoccolan, and N. C. Rust. How Does the Brain Solve Visual Object Recognition? *Neuron*, 73(3):415–434, 2012. ISSN 08966273. doi: 10.1016/j.neuron.2012.01.010.

M. Dill and S. Edelman. Imperfect invariance to object translation in the discrimination of complex shapes. *Perception*, 30(6):707–724, 2001. ISSN 03010066. doi: 10.1068/p2953.

M. G. Eley. Identifying rotated letter-like symbols. *Memory & cognition*, 10 (1):25–32, 1982. ISSN 0090-502X. doi: 10.3758/BF03197622.

C. Faubel and G. Schöner. Learning to recognize objects on the fly: a neurally based dynamic field approach. *Neural networks: the official journal of the International Neural Network Society*, 21(4):562–76, May 2008. ISSN 0893-6080. doi: 10.1016/j.neunet.2008.03.007.

C. Faubel and G. Schöner. A neuro-dynamic architecture for one shot learning of objects that uses both bottom-up recognition and top-down prediction. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*. IEEE Press, 2009.

C. Faubel and G. Schöner. Learning Objects on the Fly – Object Recognition for the Here and Now. In *Proceedings of the 2010 IEEE International Joint Conference on Neural Networks (IJCNN)*, 2010. ISBN 9781424481262.

L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003*, number 2, pages 1134–1141, 2003. ISBN 0-7695-1950-4. doi: 10.1109/ICCV.2003.1238476.

D. H. Foster and J. I. Kahn. Internal Representations and Operations in the Visual Comparison of Transformed Patterns: Effects of Pattern Point-Inversion, Positional Symmetry, and Separation. *Biological Cybernetics*, 51:305–312, 1985.

W. T. Freeman and E. H. Adelson. The Design and Use of Steerable Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9): 891–906, 1991.

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

I. Gauthier, W. G. Hayward, M. J. Tarr, A. W. Anderson, P. Skudlarski, and J. C. Gore. BOLD Activity during mental rotation and viewpoint-dependent object recognition. *Neuron*, 34(1):161–171, 2002. ISSN 08966273. doi: 10.1016/S0896-6273(02)00622-0.

M. Graf. Coordinate transformations in object recognition. *Psychological bulletin*, 132(6):920–45, November 2006. ISSN 0033-2909. doi: 10.1037/0033-2909.132.6.920.

S. Grossberg. A theory of human memory: self-organization and performance of sensory-motor codes, maps, and plans. In R. Rosen and F. Snell, editors, *Progress in Theoretical Biology*, volume 5, pages 500–639. Academic Press (Elsevier), New York, USA, 1978.

S. Grossberg and E. Mingolla. Neural Dynamics of Form Perception: Boundary Completion, Illusory Figures, And Neon Color Spreading. *Psychological Review*, 92(2):173–211, 1985. ISSN 01664115. doi: 10.1016/S0166-4115(08)61758-6.

S. Hagen, Q. C. Vuong, L. S. Scott, T. Curran, and J. W. Tanaka. The Role of Spatial Frequency in Expert Object Recognition. *Journal of Experimental Psychology: Human Perception and Performance*, 42(3):413–422, 2016. ISSN 1939-1277. doi: 10.1037/xhp0000139.

S. Han and N. Vasconcelos. Biologically plausible saliency mechanisms improve feedforward object recognition. *Vision Research*, 50(22):2295–2307, 2010. ISSN 00426989. doi: 10.1016/j.visres.2010.05.034.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*, 18(7):1527–1554, July 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527.

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154, 1962.

D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243, March 1968. ISSN 00223751. doi: 10.1113/jphysiol.1968.sp008455.

I. Iossifidis, C. Theis, C. Grote, C. Faubel, and G. Schöner. Anthropomorphism as a pervasive design concept for a robotic assistant. In *International Conference on Intelligent Robots and Systems, 2003. (IROS 2003)*, number 3, pages 3465–3472, 2003. doi: 10.1109/IROS.2003.1249692.

M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 2017–2025. Curran Associates, Inc., 2015.

J. S. Johnson, J. P. Spencer, S. J. Luck, and G. Schöner. A dynamic neural field model of visual working memory and change detection. *Psychological science*, 20(5):568–77, may 2009. ISSN 1467-9280. doi: 10.1111/j.1467-9280.2009.02329.x.

P. Jolicoeur. A size-congruency effect in memory for visual shape. *Memory & cognition*, 15(6):531–543, 1987. ISSN 0090-502X. doi: 10.3758/BF03198388.

J. I. Kahn and D. H. Foster. Visual comparison of rotated and reflected random-dot patterns as a function of their positional symmetry and separation in the field. *Quarterly Journal of Experimental Psychology Section A*, 33(2):155–166, 1981. doi: 10.1080/14640748108400782.

P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, second edition, 1999.

G. Knips, S. K. U. Zibner, H. Reimann, I. Popova, and G. Schöner. A neural dynamics architecture for grasping that integrates perception and movement generation and enables on-line updating. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 646–653. IEEE, September 2014. ISBN 978-1-4799-6934-0. doi: 10.1109/IROS.2014.6942627.

K. Koch, J. McLean, R. Segev, M. A. Freed, M. J. Berry, V. Balasubramanian, and P. Sterling. How Much the Eye Tells the Brain. *Current Biology*, 16(14):1428–1434, 2006. ISSN 09609822. doi: 10.1016/j.cub.2006.05.056.

Z. Kourtzi and C. E. Connor. Neural Representations for Object Perception: Structure, Category, and Adaptive Coding. *Annual Review*

of Neuroscience, 34(1):45–67, 2010. ISSN 0147-006X. doi: 10.1146/annurev-neuro-060909-153218.

N. Krüger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. J. Rodriguez-Sanchez, and L. Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1847–1871, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.272.

A. Larsen and C. Bundesen. Size scaling in visual pattern recognition. *Journal of experimental psychology. Human perception and performance*, 4(1): 1–20, 1978. ISSN 0096-1523. doi: 10.1037/0096-1523.4.1.1.

Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. In M. A. Arbib, editor, *The handbook of brain theory and neural networks*, volume 3361, pages 255–258. MIT Press, Cambridge, MA, USA, 1995. ISBN 0262511029. doi: 10.1109/IJCNN.2004.1381049.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

N. K. Logothetis and D. L. Sheinberg. Visual Object Recognition. *Annual Review of Neuroscience*, 19(1):577–621, March 1996. ISSN 0147-006X. doi: 10.1146/annurev.ne.19.030196.003045.

O. Lomp, S. K. U. Zibner, M. Richter, I. Rañó, and G. Schöner. A Software Framework for Cognition, Embodiment, Dynamics, and Autonomy in Robotics: Cedar. In V. Mladenov, P. Koprinkova-Hristova, G. Palm, A. E. P. Villa, B. Appollini, and N. Kasabov, editors, *Artificial Neural Networks and Machine Learning—ICANN 2013—23rd International Conference on Artificial Neural Networks*, pages 475–482, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40728-4. doi: 10.1007/978-3-642-40728-4_60.

O. Lomp, K. Terzić, C. Faubel, J. M. H. du Buf, and G. Schöner. Instance-Based Object Recognition with Simultaneous Pose Estimation Using Keypoint Maps and Neural Dynamics. In S. Wermter, C. Weber, W. Duch, T. Honkela, P. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, editors, *Artificial Neural Networks and Machine Learning—ICANN 2014: 24th International Conference on Artificial Neural Networks, Hamburg, Germany, September 15–19, 2014. Proceedings*, pages 451–458. Springer International Publishing, 2014. doi: 10.1007/978-3-319-11179-7_57.

O. Lomp, M. Richter, S. K. U. Zibner, and G. Schöner. Developing dynamic field theory architectures for embodied cognitive systems with cedar. *Frontiers in Neurorobotics*, 10, accepted 2016. ISSN 1662-5218. doi: 10.3389/fnbot.2016.00014.

O. Lomp, C. Faubel, and G. Schöner. A neural-dynamic architecture for concurrent estimation of object pose and identity. submitted 2016.

D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, nov 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94.

D. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2: 1150–1157, 1999. doi: 10.1109/ICCV.1999.790410.

J. Lücke, C. Keck, and C. von der Malsburg. Rapid convergence to feature layer correspondences. *Neural computation*, 20:2441–2463, 2008. ISSN 0899-7667. doi: 10.1162/neco.2008.06-07-539.

M. L. Mack and T. J. Palmeri. The Timing of Visual Object Categorization. *Frontiers in Psychology*, 2(July):1–8, 2011. ISSN 1664-1078. doi: 10.3389/fpsyg.2011.00165.

J. I. Markoff. Target recognition performance with chromatic and achromatic displays (Research Rep. No. SRM-148). *Honeywell, Minneapolis, MN*, 1972.

D. Marr and H. K. Nishihara. Representation and recognition of the spatial organisation of three-dimensional shapes. *Proceedings of the Royal Society of London B*, 200:269–294, 1978. ISSN 0962-8452. doi: 10.1098/rspb.1978.0020.

F. Miau, C. Papageorgiou, and L. Itti. Neuromorphic algorithms for computer vision and attention. In B. Bosacchi, D. B. Fogel, and J. C. Bezdek, editors, *Proc. SPIE 46 Annual International Symposium on Optical Science and Technology*, volume 4479, pages 12–23, Bellingham, WA, November 2001. SPIE Press. ISBN 0-8194-4193-7. doi: 10.1117/12.448343.

T. A. Nazir and K. J. O'Regan. Some results on translation invariance in the human visual system. *Spatial Vision*, 5(2):81–100, 1990.

S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100), February 1996.

B. A. Olshausen, C. H. Anderson, and D. C. Van Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *The Journal of neuroscience: the official journal of the Society for Neuroscience*, 13(11):4700–19, nov 1993. ISSN 0270-6474.

A. L. Ostergaard and J. B. Davidoff. Some effects of color on naming and recognition of objects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11(3):579–587, 1985. ISSN 1939-1285. doi: 10.1037/0278-7393.11.3.579.

J. J. Peissig and M. J. Tarr. Visual Object Recognition: Do We Know More Now Than We Did 20 Years Ago? *Annual Review of Psychology*, 58(1):75–96, 2007. ISSN 0066-4308. doi: 10.1146/annurev.psych.58.102904.190114.

M. A. Posner. Orienting of attention. *Quarterly Journal of Experimental Psychology*, 32(1):3–25, 1980. ISSN 0033-555X. doi: 10.1080/00335558008248231.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*, volume 29. second edition, 1992. ISBN 0521431085. doi: 10.2307/1269484.

M. Richter, Y. Sandamirskaya, and G. Schöner. A robotic action selection and behavioral organization architecture inspired by human cognition. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2457–2464, 2012. ISSN 2153-0858.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999. ISSN 1097-6256. doi: 10.1038/14819.

M. Riesenhuber and T. A. Poggio. How Visual Cortex Recognizes Objects: The Tale of the Standard Model. In L. M. Chapula and J. S. Werner, editors, *The visual neurosciences*, pages 1640–1653. Cambridge, MA: MIT Press, 2003. ISBN 0262033089.

J. Rodrigues and J. M. H. du Buf. Multi-scale lines and edges in V1 and beyond: Brightness, object categorization and recognition, and consciousness. *BioSystems*, 95(3):206–226, 2009. ISSN 03032647. doi: 10.1016/j.biosystems.2008.10.006.

B. Rossion and G. Pourtois. Revisiting Snodgrass and Vanderwart's object pictorial set: The role of surface detail in basic-level object recognition. *Perception*, 33(2):217–236, 2004. ISSN 0301-0066. doi: 10.1068/p5117.

J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *arXiv preprint arXiv:1404.7828*, 61:1–66, 2014. ISSN 08936080. doi: 10.1016/j.neunet.2014.09.003.

S. Schneegans and G. Schöner. A neural mechanism for coordinate transformation predicts pre-saccadic remapping. *Biological Cybernetics*, 106(2): 89–109, 2012. ISSN 03401200. doi: 10.1007/s00422-012-0484-8.

G. Schöner. Dynamical Systems Approaches to Cognition. In R. Sun, editor, *The Cambridge Handbok of Computational Psychology*, chapter 4, pages 101–126. Cambridge University Press, 2008.

G. Schöner, C. Faubel, E. Dineva, and E. Bicho. Embodied neural dynamics. In *Dynamic Thinking: A Primer on Dynamic Field Theory*, pages 95–118. 2015a. ISBN 9780199300563.

G. Schöner, J. Spencer, and The DFT Research Group. *Dynamic Thinking: A Primer on Dynamic Field Theory*. Oxford University Press, first edition, 2015b. ISBN 9780199300563.

T. Serre, L. Wolf, and T. Poggio. A new biologically motivated framework for robust object recognition. Technical report, DTIC Document, 2004.

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE transactions on pattern analysis and machine intelligence*, 29(3):411–26, March 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.56.

R. N. Shepard and J. Metzler. Mental Rotation of Three-Dimensional Objects. *Science New Series*, 171(3972):701–703, 1971.

D. J. Simons and C. F. Chabris. Gorillas in our midst: Sustained inattentional blindness for dynamic events. *Perception*, 28(9):1059–1074, 1999. ISSN 03010066. doi: 10.1068/p2952.

D. J. Simons and D. T. Levin. Change blindness. *Trends in cognitive sciences*, 1(7):261–267, 1997. ISSN 1364-6613. doi: 10.1016/S1364-6613(97)01080-2.

M. J. Tarr. Rotating objects to recognize them: A case study on the role of viewpoint dependency in the recognition of three-dimensional objects.

Psychonomic Bulletin & Review, 2(1):55–82, 1995. ISSN 1069-9384. doi: 10.3758/BF03214412.

M. J. Tarr and S. Pinker. Mental rotation and orientation-dependence in shape recognition. *Cognitive Psychology*, 21(2):233–282, apr 1989. ISSN 00100285. doi: 10.1016/0010-0285(89)90009-1.

K. Terzić, J. M. F. Rodrigues, and J. M. H. du Buf. Fast Cortical Keypoints for Real-Time Object Recognition. In *2013 IEEE International Conference on Image Processing*, pages 3372–3376. IEEE, 2013.

N. J. Thomas. Mental imagery. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2016 edition, 2016.

S. J. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system, 1996. ISSN 0028-0836.

S. Ullman. Aligning pictorial descriptions: an approach to object recognition. *Cognition*, 32(3):193–254, 1989. ISSN 00100277. doi: 10.1016/0010-0277(89)90036-X.

P. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:511–518, 2001. ISSN 1063-6919. doi: 10.1109/CVPR.2001.990517.

R. Vogels. Categorization of complex visual images by rhesus monkeys. Part 2: single-cell study. *European Journal of Neuroscience*, 11(4):1239–1255, 1999. ISSN 0953816X. doi: 10.1046/j.1460-9568.1999.00530.x.

D. Walther and C. Koch. Modeling attention to salient proto-objects. *Neural Networks*, 19(9):1395–1407, 2006. ISSN 08936080. doi: 10.1016/j.neunet.2006.10.001.

D. B. Walther and C. Koch. Attention in hierarchical models of object recognition. *Progress in brain research*, 165(06):1–38, 2007. doi: 10.1016/S0079-6123(06)65005-X.

B. A. Wandell. *Foundations of Vision*. Sinauer Associates, 1995.

L. Wiskott and T. J. Sejnowski. Slow feature analysis: unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002. ISSN 0899-7667. doi: 10.1162/089976602317318938.

P. Wolfrum, C. Wolff, J. Lücke, and C. von der Malsburg. A recurrent dynamic model for correspondence-based face recognition. *Journal of Vision*, 8(7), 2008. ISSN 1534-7362. doi: 10.1167/8.7.34.

L. H. Wurm, G. E. Legge, L. M. Isenberg, and A. Luebker. Color improves object recognition in normal and low vision. *Journal of Experimental Psychology. Human Perception and Performance*, 19(4):899–911, 1993. ISSN 0096-1523. doi: 10.1037/0096-1523.19.4.899.

S. K. U. Zibner, C. Faubel, I. Iossifidis, and G. Schöner. Scene Representation for Anthropomorphic Robots: A Dynamic Neural Field Approach. *ISR / Robotik 2010*, pages 927–933, 2010.

S. K. U. Zibner, C. Faubel, and G. Schöner. Making a robotic scene representation accessible to feature and label queries. *2011 IEEE International Conference on Development and Learning, ICDL 2011*, 2011a. ISSN 2161-9476. doi: 10.1109/DEVLRN.2011.6037360.

S. K. U. Zibner, C. Faubel, I. Iossifidis, and G. Schöner. Dynamic Neural Fields as Building Blocks of a Cortex-Inspired Architecture for Robotic Scene Representation. *IEEE Transactions on Autonomous Mental Development*, 3(1):74–91, 2011b. ISSN 1943-0604. doi: 10.1109/TAMD.2011.2109714.

S. K. Zibner and C. Faubel. Dynamic Scene Representations and Autonomous Robotics. In *Dynamic Thinking: A Primer on Dynamic Field Theory*, chapter 9, pages 227–246. Oxford University Press, 2015. ISBN 9780199300563.

D. Zwillinger. *Handbook of differential equations*. Academic Press, San Diego, CA, USA, 1989.

Part V

Appendix

Appendix A

Index of notation

symbol	brief description	page
$G_\sigma(x)$	A Gaussian function centered around zero.	8
$G_{\sigma,\mathbf{c}}(\mathbf{x})$	A Gaussian function centered around $\mathbf{c}$.	39
$g(u)$	Sigmoid (logistic) function.	8
$H(u)$	Heaviside (step) function.	8
$\sigma_+(u)$	Semi-linear transfer function.	36
$f * g$	Convolution of the functions f and g.	26
$[f * g](x,t)$	Result of convolution of the functions f and g at location x and time, t. Time is disregarded in the convolution operation.	26
$f \circ g$	Concatenation of two functions, i.e., $f(g(\cdot))$.	26
$(f \circ g)(x)$	Concatenation of two functions, i.e., $f(g(x))$.	26
$\hat{F}$	Normalized, mean-free variant of a function, F.	33

Appendix B

Clustering peaks

Reading out the representation in a three-dimensional space-feature working memory field requires determining the centers of peaks in the field. In practice, I first project the field to the spatial dimensions and then apply a clustering algorithm to find the peak centers in the reduced representation.

Algorithm 1 describes the method I use for the clustering peaks. Its input is a matrix representing the discretized result of the projection to the spatial dimensions. The algorithm clusters neighboring suprathreshold entries in this matrix into a single peak. This process is started by the outer while loop: the suprathreshold location with the highest value that has not yet been visited is selected and added to the set, X, of locations to explore for a new peak. In the inner while loop, the algorithm picks a location from X and iterates over its eight direct neighbors. The ones that are above threshold are added to the current peak and to the set of points to be explored. The inner loop continues until no more suprathreshold neighbors are available, yielding a set L that contains all matrix entries considered to be part of the current peak. The location of the peak is calculated as the average of these locations. The outer loop repeats this process until all suprathreshold locations have been visited.

input : A discrete matrix, $\sigma_{l,m}$, where $0 \leq l < n_l$ and $0 \leq m < n_m$ refer to the discretized spatial dimension of the field; a threshold $\theta > 0$.
output: A set, P, of peaks, $p_k = (x_k, y_k)$ where $x_k, y_k \in \mathbb{R}$ are the estimated peak centers.

```
//m_{a,b} is a structure used for marking locations that have
  already been assigned to a peak.
```
$m_{a,b} \leftarrow 0 \; \forall 0 \; \leq a < n_i, 0 \leq b < n_j$;
$P \leftarrow \emptyset$;
```
//Find maximally active location that is not yet marked as
  part of a peak
```
while $\exists i, j : \; m_{i,j} = 0 \wedge \sigma_{i,j} \leq \sigma_{i',j'} \; \forall i', j' \wedge \sigma_{i,j} > \theta$ **do**
  $X \leftarrow \{(i,j)\}$; `//Points to explore`
  $L \leftarrow \emptyset$; `//Points that are part of the current peak`
  while $X \neq \emptyset$ **do**
    take (i_x, j_x) from X;
    $X \leftarrow X \setminus (i_x, j_x)$;
    $L \leftarrow L \cup \{(i_x, j_x)\}$;
    $m_{i_x,j_x} \leftarrow 1$;
    `//Iterate over neighbors, add suprathreshold ones to the peak and queue them to be explored as well.`
    for $i' \leftarrow -1$ **to** 1 **do**
      for $j' \leftarrow -1$ **to** 1 **do**
        if $m_{i_x+i',j_x+j'} > \theta \wedge 0 \leq i_x + i' < n_i \wedge 0 \leq j_x + j' < n_j$ **then**
          $X \leftarrow X \cup (i_x + i', j_x + j')$;
          $L \leftarrow L \cup (i_x + i', j_x + j')$;
        end
      end
    end
  end
  `//Use the mean of the points in` L `as the position of the peak.`
  $i_l \leftarrow \frac{1}{|L|} \sum_{(i',j') \in L} i', \quad j_l \leftarrow \frac{1}{|L|} \sum_{(i',j') \in L} j'$;
  $P \leftarrow P \cup (i_l, j_l)$
end

Algorithm 1: The algorithm used for clustering peak centers.

Appendix C

Mapping between keypoint scales and scale estimates

In Part III, I describe contributions to scale estimation from keypoint scales as well as selection of a keypoint scale based on a scale estimate. However, the scale estimates and keypoint scales are defined over different coordinate systems, and the nontrivial transformation between them must be taken into account in implementation.

Keypoints have a scale σ which is proportional to the wavelength of the corresponding Gabor wavelets, λ (see Chapter 5). The coordinate system for the scale estimates, on the other hand, is defined by the log-polar transformation used for the scale transformation. In my implementation, the log-polar transformation of a pattern, $P : \mathbb{R}^2 \to \mathbb{R}$, is given by

$$P_{\mathrm{lp}}(\rho, \phi) = P\left(\exp\left(\frac{\rho}{m}\right)\ \sin(\phi), \exp\left(\frac{\rho}{m}\right)\ \cos(\phi)\right), \tag{C.1}$$

where m is a magnitude parameter (used to control sampling effects in the discretized version of the transformation) and the index 'lp' makes explicit that the pattern is in log-polar space. For simplicity of notation, I assume that the origin of the transformation is the origin of the pattern. The inverse of this transformation is thus given by

$$P(x, y) = P_{\mathrm{lp}}\left(m\ \log\sqrt{x^2 + y^2}, \arctan(y, x)\right). \tag{C.2}$$

To determine how the scale estimates and match value from a specific keypoint scale are related, the coordinate system of the fields representing scale must made explicit. Because scale is applied by convolving with the scale estimate, it is implied that the dimension is defined such that the origin of its coordinate system corresponds to a scale factor of one (transforming

by a scale estimate that is one only at $\rho = 0$ and zero everywhere else corresponds to an identity transformation). Let δ_ρ be a position in this coordinate system. Scaling the pattern, P, by δ_ρ in log-polar space means shifting its argument by this value. The scaled pattern is therefore given by

$$P_{\text{lp}}^{\text{sc}}(\rho, \phi) = P_{\text{lp}}(\rho - \delta_\rho, \phi) \tag{C.3}$$

The resulting scale factor in Cartesian space can be obtained by transforming the pattern back to this space. Substituting $\rho' = m \log \sqrt{x^2 + y^2}$ and $\phi' = \arctan(y, x)$, the scaled pattern in Cartesian space is

$$P^{\text{sc}}(x', y') = P_{\text{lp}}^{\text{sc}}(\rho', \phi') = P_{\text{lp}}(\rho' - \delta_\rho, \phi'). \tag{C.4}$$

By using the transformation in Equation C.1, this becomes

$$P^{\text{sc}}(x', y') = P\left(\exp\left(\frac{\rho' - \delta_\rho}{m}\right)\ \sin(\phi'),\ \exp\left(\frac{\rho' - \delta_\rho}{m}\right)\cos(\phi')\right). \tag{C.5}$$

From this, it follows that the relation between the x' coordinate of the scaled pattern and that of the original pattern is given as

$$\begin{aligned} x' &= \exp\left(\frac{\rho' - \delta_\rho}{m}\right)\ \sin(\phi') \\ &= \exp\left(\frac{\rho'}{m}\right)\ \exp\left(-\frac{\delta_\rho}{m}\right)\ \sin(\phi'). \end{aligned} \tag{C.6}$$

Inserting the definition of ρ', this becomes

$$\begin{aligned} x' &= \exp\left(\frac{m\ \log\sqrt{x^2 + y^2}}{m}\right)\ \exp\left(-\frac{\delta_\rho}{m}\right)\ \sin(\phi') \\ &= \exp\left(-\frac{\delta_\rho}{m}\right)\ \sqrt{x^2 + y^2}\ \sin(\phi') \\ &= \exp\left(-\frac{\delta_\rho}{m}\right)\ x. \end{aligned} \tag{C.7}$$

For the second argument of the pattern, the relationship $y' = \exp\left(-\frac{\delta_\rho}{m}\right)\ y$ can be found using analogous derivations. This means that a shift of δ_ρ in log-polar space scales the pattern by a factor of $f = \exp\left(-\frac{\delta_\rho}{m}\right)$.

With this knowledge, the function $m_s(\sigma) = \delta_\rho$ required for mapping a scale estimate to an input scale (introduced in Section 12.1.1) can be derived. First, the input scales, σ, which usually refer to the size of a Gabor wavelet,

must be converted to a scale factor. Since the scale estimate is defined relative to the training image, the same may be done for the input scales. Thus, $f_\sigma = \frac{\sigma}{\sigma_{\text{train}}}$ (where σ_{train} is the scale on which the system was trained). Because both δ_ρ and f_σ are required to be equal for the mapping function m_s to be valid, it then follows that

$$f_\sigma = \frac{\sigma}{\sigma_{\text{train}}} = \exp\left(-\frac{\delta_\rho}{m}\right). \tag{C.8}$$

Solving for δ_ρ yields the mapping function,

$$m_s(\sigma) = \delta_\rho = -m \ \log \frac{\sigma}{\sigma_{\text{train}}}. \tag{C.9}$$