Linköping University Post Print

Efficient computation of channel-coded feature maps through piecewise polynomials

Erik Jonsson and Michael Felsberg

N.B.: When citing this work, cite the original article.

Original Publication:

Erik Jonsson and Michael Felsberg, Efficient computation of channel-coded feature maps through piecewise polynomials, 2009, Image and Vision Computing, (27), 11, 1688-1694.

<http://dx.doi.org/10.1016/j.imavis.2008.11.002>

Copyright: Elsevier Science B.V., Amsterdam.

<http://www.elsevier.com/>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-21197>

Efficient Computation of Channel-coded Feature Maps through Piecewise Polynomials

Erik Jonsson and Michael Felsberg

*Computer Vision Laboratory
Dept. of Electrical Engineering, Linköping*

Abstract

Channel-coded feature maps (CCFMs) represent arbitrary image features using multi-dimensional histograms with soft and overlapping bins. This representation can be seen as a generalization of the SIFT descriptor, where one advantage is that it is better suited for computing derivatives with respect to image transformations. Using these derivatives, a local optimization of image scale, rotation and position relative to a reference view can be computed. If piecewise polynomial bin functions are used, e.g. B-splines, these histograms can be computed by first encoding the data set into a histogram-like representation with non-overlapping multidimensional monomials as bin functions. This representation can then be processed using multi-dimensional convolutions to obtain the desired representation. This allows to reuse much of the computations for the derivatives. By comparing the complexity of this method to direct encoding, it is found that the piecewise method is preferable for large images and smaller patches with few channels, which makes it useful, e.g. in early steps of coarse-to-fine approaches.

1 Introduction

Channel-coded feature maps (CCFMs) are a general way of representing image features like color and gradient orientation using channel representation techniques [5]. The basic idea is to create a joint histogram of spatial position and feature value, as illustrated in Fig. 1. This can be viewed as a number of local feature histograms for different spatial positions. The most well-known

¹ The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 215078 DIPLECS and from the European Community's Sixth Framework Programme (FP6/2003-2007) under grant agreement n° 004176 COSPAL.

Fig. 1. Illustration of a Channel-coded feature map. For each spatial channel, there is a soft histogram of chromacity and orientation, giving in total a 4D histogram.

variety of this kind of representations is the SIFT descriptor [7], where position and edge orientation are encoded into a three-dimensional histogram. While SIFT uses linear interpolation to assign points to several bins, in general more advanced interpolations can be used, which can be modeled as having some basis function sitting at each bin. If we want to adjust the position in an image at which the CCFM is computed, e.g. in order to track an object between frames, the derivatives of the CCFM with respect to scale change, rotation and translation are needed. These derivatives are more well-behaved if the basis functions are smooth.

Various forms of feature histograms are rather common in object recognition and tracking. The *shape contexts* used in [1] are log-polar histograms of edge point positions. Since they are only used as a cue for point matching, no attempt of computing their derivatives with respect to image transformations is made. In [2], objects are tracked using single (global) color histograms, weighted spatially with a smooth kernel. This can be seen as using a channel-coded feature map with only a single spatial channel. The gradient-based optimization is restricted to translations, scale changes are handled by testing a number of discrete scales, and rotation is not handled at all. In [12], orientation histograms and downsampled color images are constructed efficiently by box filters using integral images [11]. This is possible since rotation of the tracked object is not considered. Usually when SIFT features are used in tracking (e.g. in [8]), the descriptors are computed at fixed positions as originally described in [7] without attempting to fine-tune the similarity parameters.

The channel-coded feature maps generalize all these approaches, allow for arbitrary basis functions, support derivatives with respect to rotation, translation and scale changes, but are comparably time consuming to compute. While it is possible to track a single CCFM in almost real-time even using a straightforward encoding [6], time becomes a limiting factor when several objects are considered simultaneously. This motivates looking for efficient algorithms. When the basis functions are piecewise polynomials, so are their derivatives, which indicates that some time may be saved by exploiting this piecewiseness. The key contribution of this paper is exploring this property and present-

ing an algorithm for computing CCFMs and their derivatives using piecewise polynomials. This unfolds as a rather involved exercise in multi-dimensional convolutions and index book-keeping, and in order to write everything down in a convenient way some new notation is introduced. The computational complexity of this approach is compared to a more straight-forward algorithm, and situations where the piecewise approach is beneficial are identified.

Section 2 describes and defines channel-coded feature maps and how to optimize the similarity parameters. Section 3 contains the main contribution of this paper - how to compute the feature maps including their derivatives using piecewise polynomials. In Sect. 4, the computational complexity of the proposed method is analyzed theoretically, and the running times of two competing implementations are compared empirically.

2 Channel-coded Feature Maps

2.1 Introductory Example

A channel-coded feature map (CCFM) is a soft histogram with overlapping bin functions in the joint space of spatial coordinates and feature values, referred to as the *spatio-featural space* (SF-space). Assuming that we have two feature maps $z_i = z(u, v)$ and $q_i = q(u, v)$, giving e.g. the chromacity and gradient orientation for every pixel (u_i, v_i) , the channel-coded feature map is

$$\mathbf{c}[\tilde{u}, \tilde{v}, \tilde{z}, \tilde{q}] = \sum_i B(u_i - \tilde{u}, v_i - \tilde{v}, z_i - \tilde{z}, q_i - \tilde{q}) . \quad (1)$$

The C/Java-inspired bracket is used to denote vector and array elements in order to avoid overloading the notation with too many sub- and superscripts. The vector $\tilde{\mathbf{x}} = [\tilde{u}, \tilde{v}, \tilde{z}, \tilde{q}]$ is called a *channel center*. Note that $\mathbf{c}$ is to be interpreted as a multi-dimensional array of numbers, meaning that this sum is only evaluated for channel centers located on a regular grid. For the sake of presentation, we consider only centers that are unit-spaced in each dimension, such that $\tilde{u}, \tilde{v}, \tilde{z}, \tilde{q}$ are integers. This simplifies the notation since the symbol used for denoting channel centers can be reused as index into $\mathbf{c}$, and can be done without loss of generality since it is only a matter of rescaling the input space.

The basis function (or bin function) B is some bell-shaped smooth function with a compact support that extends past $[-0.5, 0.5]$ in each dimension, such that the *receptive fields* of neighboring channels are overlapping. In this paper, we will only consider separable basis functions composed of splines [10] in each dimension. For our purposes, splines should be understood as piecewise

Fig. 2. Two neighboring spline bin functions in one dimension with the knots marked. Left: First order B-spline. Right: Second order B-spline.

polynomial functions with unit-spaced knots, not necessarily fulfilling some smoothness constraints (like being k times differentiable). Some examples of plausible spline basis functions are shown in Fig. 2.

2.2 Motivation

Creating a channel-coded feature map from an image is a way of obtaining a coarse spatial resolution but still maintaining much information at each position. A 128×128 grayscale image can be converted to e.g. a 12×12 patch with 8 layers, where each layer represents the presence of a certain orientation. This is advantageous for methods that adjust the spatial location of an image patch based on a local optimization in the spirit of the KLT tracker [9]. The low spatial resolution increases the probability of reaching the correct optimum of the energy function, while having more information at each pixel improves the robustness and accuracy. Furthermore, using smooth basis functions instead of hard histograms makes the derivatives of the representation well-behaved.

Note that if we use weighted local gradient orientation as a single image feature, and use first-order B-splines as basis functions, the channel-coded feature map becomes equivalent to the SIFT *descriptor* [7] (referring to the final step of the SIFT procedure). The SIFT descriptor typically contains a 128-dimensional vector, which is computed in the same way as a first-order B-spline channel-coded feature map with four channels in each spatial dimension and eight channels in the orientation dimension. The only difference is that the SIFT operator additionally weights the samples added to the bins by a Gaussian function. Hence, the algorithm presented in this paper can be used for computing SIFT descriptors and their derivatives with respect to scale, orientation, and spatial position efficiently. Replacing the original binning algorithm of the SIFT descriptor extraction, which corresponds to what we later call *direct* encoding, with the proposed *piecewise* method, results in an algorithm that computes SIFT descriptors *including their derivatives* just 25% slower than the original SIFT algorithm. The benefit of the derivatives are twofold: They can be used for fitting local frames (tracking) and as a stability criterion of the SIFT descriptor.

2.3 General Definition and Notation

In general, we consider an arbitrary number of feature dimensions. Let $\mathcal{X} = \{\mathbf{x}_i\}_i$ be a set of points in a D-dimensional spatio-feature space $\mathcal{F}$, where the first two elements of each $\mathbf{x}_i$ are the spatial image positions and the rest are feature values. Let $\tilde{\mathbf{x}} \in \mathcal{F}$ be a channel center or, equivalently, a D-dimensional vector of integer indices. Let B be a D-dimensional separable spline. The CCFM of $\mathcal{X}$ is then a multi-dimensional array

$$\mathbf{c}[\tilde{\mathbf{x}}] = \frac{1}{I} \sum_i w_i B(\mathbf{x}_i - \tilde{\mathbf{x}}) . \quad (2)$$

Sometimes it will be practical to denote the two first elements of $\mathbf{x}$ as u and v to stress that they are in fact image coordinates, i.e. we define $u = \mathbf{x}[1]$, $v = \mathbf{x}[2]$, and similarly $\tilde{u} = \tilde{\mathbf{x}}[1]$, $\tilde{v} = \tilde{\mathbf{x}}[2]$.

2.4 Local Optimization of Similarity Parameters

One issue in applications like object pose estimation, tracking and image registration is the fine-tuning of similarity parameters. The problem is to find a similarity transform that maps one image (or image patch) to another image in a way that minimizes some distance function. One way to solve this is to encode the first image or patch into a *target CCFM* $\mathbf{c}_0$. Let $f(s, \alpha, b_u, b_v)$ be a function that extracts a CCFM from a patch in the second image (the query image) of radius² s and rotation α , located at (b_u, b_v) . We then look for the parameters that make $e = \|f(s, \alpha, b_u, b_v) - \mathbf{c}_0\|^2$ minimal. In [6], this problem is one component of an appearance-based pose estimation method, and the optimization problem is solved using a Gauss-Newton method.

In order to minimize e with a local optimization, the derivatives of $f(s, \alpha, b_u, b_v)$ with respect to the similarity parameters s, α, b_u, b_v are needed. From [6] equations (25) to (28) (changing the notation to fit the current presentation), these

² Perpendicular distance between center and border

derivatives are

$$\frac{d\mathbf{c}[\tilde{\mathbf{x}}]}{db_u} = -\frac{1}{I} \sum_i w_i B'_u(\mathbf{x}_i - \tilde{\mathbf{x}}) \quad (3)$$

$$\frac{d\mathbf{c}[\tilde{\mathbf{x}}]}{db_v} = -\frac{1}{I} \sum_i w_i B'_v(\mathbf{x}_i - \tilde{\mathbf{x}}) \quad (4)$$

$$\frac{d\mathbf{c}[\tilde{\mathbf{x}}]}{ds} = -\frac{1}{I} \sum_i w_i \left(2B(\mathbf{x}_i - \tilde{\mathbf{x}}) + u_i B'_u(\mathbf{x}_i - \tilde{\mathbf{x}}) + v_i B'_v(\mathbf{x}_i - \tilde{\mathbf{x}}) \right) \quad (5)$$

$$\frac{d\mathbf{c}[\tilde{\mathbf{x}}]}{d\alpha} = \frac{1}{I} \sum_i w_i \left(v_i B'_u(\mathbf{x}_i - \tilde{\mathbf{x}}) - u_i B'_v(\mathbf{x}_i - \tilde{\mathbf{x}}) \right) \quad (6)$$

When constructing an algorithm for computing CCFMs, these derivatives should also be considered. In the proposed algorithm, the first step in computing the original encoding can be reused in computing the derivatives, leading to an efficient method.

3 Implementation Using Piecewise Polynomials

3.1 Monopieces and Polypuzzles

In [10], k 'th order splines are uniquely characterized by an expansion in shifted k 'th order B-splines. Here we take a different approach. Note that a polynomial is characterized by a coefficient for each participating monomial, and a piecewise polynomial is characterized by a set of such coefficients for each piece. In order to express this compactly, a one-dimensional *monopiece* $P^{(p)}(x)$ of order p is introduced as

$$P^{(p)}(x) = \begin{cases} x^p & \text{if } -0.5 < x \leq 0.5 \\ 0 & \text{otherwise} \end{cases} . \quad (7)$$

Using these monopieces, any piecewise polynomial function with unit-spaced knots can be written as

$$B(x) = \sum_p \sum_s \mathbf{K}[p, s] P^{(p)}(x + s) . \quad (8)$$

Such a function $B(x)$ will be called a (one-dimensional) *polypuzzle*. In contrast to splines as defined in [10], this definition also support non-smooth and even non-continuous piecewise polynomials. In practice however, only continuous functions will be considered - otherwise the derivatives are not well-defined. As long as the knots are unit-spaced, all splines are polypuzzles.

The shifts s can be at the integers or offset by 0.5, which is the case e.g. for odd-order centralized B-splines. $\mathbf{K}$ is a matrix holding coefficients for each polynomial order and shift. Note that we index this matrix directly with p and s even though s is not necessarily an integer. Think of $\mathbf{K}$ as a mapping $\mathcal{P} \times \mathcal{S} \rightarrow \mathbb{R}$, where $\mathcal{P}$ is the set of polynomial orders and $\mathcal{S}$ is the set of shifts used. These sets are finite, so $\mathbf{K}$ can be represented by a matrix. Some examples are given in Table 1.

The derivative of a continuous polypuzzle is another polypuzzle of lower order, and it is simple to state the relationship between the coefficient matrices of the two. Let $\mathbf{K}'$ be the coefficient matrix of B' . From the rules of polynomial differentiation, it follows that $\mathbf{K}'[p, s] = (p + 1)\mathbf{K}[p + 1, s]$.

3.2 Multidimensional Monopieces

A *multi-dimensional monopiece* is a separable function consisting of one monopiece in each dimension:

$$P^{(\mathbf{p})}(\mathbf{x}) = \prod_d P^{(\mathbf{p}[d])}(\mathbf{x}[d]) . \quad (9)$$

The vector $\mathbf{p}$ defines the polynomial order in each dimension and will be called a *polynomial order map*. A multi-dimensional separable polypuzzle is

$$B(\mathbf{x}) = \prod_d B_d(\mathbf{x}[d]) . \quad (10)$$

Combining (10) with (8) gives the multi-dimensional polypuzzle expressed in terms of monopieces:

$$B(\mathbf{x}) = \prod_d \sum_p \sum_s \mathbf{K}_d[p, s] P^{(p)}(\mathbf{x}[d] + s) . \quad (11)$$

We expand this product and note that each term contains a product of one single-dimensional monopiece in each dimension according to

$$B(\mathbf{x}) = \sum_{s_1} \cdots \sum_{s_D} \sum_{p_1} \cdots \sum_{p_D} \prod_d \mathbf{K}_d[p_d, s_d] P^{(p_d)}(\mathbf{x}[d] + s_d) . \quad (12)$$

We can now gather all shift and polynomial order indices into index vectors $\mathbf{s}$ and $\mathbf{p}$. The coefficients in front of the monopieces can be combined into a multi-dimensional version of $\mathbf{K}$ defined as³

$$\mathbf{K}[\mathbf{p}, \mathbf{s}] = \prod_d \mathbf{K}_d[\mathbf{p}[d], \mathbf{s}[d]] . \quad (13)$$

³ We reuse the symbol $\mathbf{K}$ for the multi-dimensional case. The two meanings will be disambiguated by the arguments of $\mathbf{K}$.

Furthermore, the product of the monopieces can be written as a multidimensional monopiece $P^{(\mathbf{p})}(\mathbf{x} - \mathbf{s})$, which lets us write (12) more compactly as

$$B(\mathbf{x}) = \sum_{\mathbf{p}} \sum_{\mathbf{s}} \mathbf{K}[\mathbf{p}, \mathbf{s}] P^{(\mathbf{p})}(\mathbf{x} + \mathbf{s}) . \quad (14)$$

This equation simply states that the multi-dimensional polypuzzle can be written as a linear combination of translated multi-dimensional monopieces of different orders. The coefficients in front of the monopieces are organized in an array $\mathbf{K}$, giving the coefficient for each translation $\mathbf{s}$ and polynomial order map $\mathbf{p}$.

3.3 Monopiece Encodings

The proposed way of computing the channel-coded feature maps is to go via a *monopiece encoding*, defined as

$$\mathbf{c}_{\text{mp}}[\mathbf{p}, \tilde{\mathbf{x}}] = \frac{1}{I} \sum_i w_i P^{(\mathbf{p})}(\mathbf{x}_i - \tilde{\mathbf{x}}) . \quad (15)$$

This can be seen as a multi-layered CCFM where each $\mathbf{p}$ -layer corresponds to one polynomial order configuration $\mathbf{p}$. Each layer is like a histogram using $P^{(\mathbf{p})}$ as basis function instead of the box function. The channels with different centers $\tilde{\mathbf{x}}$ within each layer are not overlapping, meaning that each pixel of the input image belongs to exactly one channel per layer. To compute this encoding, simply loop over the image, compute the value of each polynomial for each pixel, and accumulate the result into $\mathbf{c}$.

This is a generalization of the P-channel encoding used in [3, 4], where only one constant and one linear function in each dimension were used. Here, we will use all monopieces needed for our target B-spline encoding. Which these are will be stated in Sect. 3.7.

3.4 Converting Monopiece Encodings to CCFMs

Recall the definition of channel-coded feature maps from (2), repeated here for convenience:

$$\mathbf{c}[\tilde{\mathbf{x}}] = \frac{1}{I} \sum_i w_i B(\mathbf{x}_i - \tilde{\mathbf{x}}) . \quad (16)$$

In order to see how to get from the monopiece encoding in (15) to the CCFM in (16), we rewrite (16) in terms of monopieces according to (14), rearrange

the sums, and plug in the definition of the monopiece encodings from (15):

$$\begin{aligned}
\mathbf{c}[\tilde{\mathbf{x}}] &\stackrel{(14)}{=} \frac{1}{I} \sum_i w_i \left[\sum_{\mathbf{p}} \sum_{\mathbf{s}} \mathbf{K}[\mathbf{p}, \mathbf{s}] P^{(\mathbf{p})}(\mathbf{x}_i - \tilde{\mathbf{x}} + \mathbf{s}) \right] = \\
&= \sum_{\mathbf{p}} \sum_{\mathbf{s}} \mathbf{K}[\mathbf{p}, \mathbf{s}] \left[\frac{1}{I} \sum_i w_i P^{(\mathbf{p})}(\mathbf{x}_i - \tilde{\mathbf{x}} + \mathbf{s}) \right] = \\
&\stackrel{(15)}{=} \sum_{\mathbf{p}} \sum_{\mathbf{s}} \mathbf{K}[\mathbf{p}, \mathbf{s}] \mathbf{c}_{\text{mp}}[\mathbf{p}, \tilde{\mathbf{x}} - \mathbf{s}] .
\end{aligned} \tag{17}$$

At this point, it is convenient to use the $(\cdot)$ -notation. With $\mathbf{c}_{\text{mp}}[\mathbf{p}, \cdot]$, we mean a function $\tilde{\mathbf{x}} \mapsto \mathbf{c}_{\text{mp}}[\mathbf{p}, \tilde{\mathbf{x}}]$. Similarly, $\mathbf{K}[\mathbf{p}, \cdot]$, is a function $\tilde{\mathbf{x}} \mapsto \mathbf{K}[\mathbf{p}, \tilde{\mathbf{x}}]$. Since these are functions of a vector variable, we can apply a multi-dimensional discrete convolution operator according to:

$$(\mathbf{K}[\mathbf{p}, \cdot] * \mathbf{c}_{\text{mp}}[\mathbf{p}, \cdot])(\tilde{\mathbf{x}}) = \sum_{\mathbf{s}} \mathbf{K}[\mathbf{p}, \mathbf{s}] \mathbf{c}_{\text{mp}}[\mathbf{p}, \tilde{\mathbf{x}} - \mathbf{s}] . \tag{18}$$

This is recognized from the right-hand side of (17), which to summarize gives us

$$\mathbf{c} = \sum_{\mathbf{p}} \mathbf{K}[\mathbf{p}, \cdot] * \mathbf{c}_{\text{mp}}[\mathbf{p}, \cdot] . \tag{19}$$

In words, in order to convert from a monopiece encoding to a CCFM, we must perform one multidimensional convolution in each $\mathbf{p}$ -layer, and then sum over all layers. Note from the definition of $\mathbf{K}$ in (13) that for any given $\mathbf{p}$, the filter kernel $\mathbf{K}[\mathbf{p}, \cdot]$ is separable. This means that each convolution in (19) can be computed as a sequence of one-dimensional convolutions with kernel $\mathbf{K}_d[\mathbf{p}[d], \cdot]$ in dimension d . The complete algorithm is summarized as Alg. 1.

Algorithm 1 Creating CCFMs from monopiece encodings.

Inputs:

- Monopiece encoding $\mathbf{c}_{\text{mp}}$
- Kernel definition matrix $\mathbf{K}_d$ for each dimension d

Initialization:

- **result** = Zero N-dimensional array

```

for each polynom order map  $\mathbf{p}$  do
  thislayer =  $\mathbf{c}_{\text{mp}}[\mathbf{p}, \cdot]$ 
  for each dimension  $d$  do
    kernel1d =  $\mathbf{K}_d[\mathbf{p}[d], \cdot]$ 
    Reshape kernel1d to be along dimension  $d$ 
    thislayer = thislayer * kernel1d (1D convolution)
  end for
  result += thislayer
end for

```

3.5 Generalized Notation

In the next section, we describe how the derivatives are computed in the piecewise polynomial framework. In order to do that, we first need to further extend our notation. Recall that each multi-dimensional polypuzzle is characterized by its coefficient array $\mathbf{K}$. In order to be able to express such matrices for different functions conveniently, we define $\mathbf{K}\{f\}[\mathbf{p}, \tilde{\mathbf{x}}]$ for any polypuzzle f such that

$$f(\mathbf{x}) = \sum_{\mathbf{p}, \mathbf{s}} \mathbf{K}\{f\}[\mathbf{p}, \mathbf{s}] P^{(\mathbf{p})}(\mathbf{x} + \mathbf{s}) . \quad (20)$$

To be strict, there should always be a *function* within the $\{\}$ -brackets, but the notation is simplified slightly by allowing ourselves to write things like $\mathbf{K}\{vB'_u\}$, to be interpreted as $\mathbf{K}\{\mathbf{x} \mapsto vB'_u(\mathbf{x})\}$, where $\mathbf{x} = [u, v, z_1, z_2, \dots]^T$ as in the previous chapter. Furthermore, we denote different complete encodings as $\mathbf{c}\{f\}$, meaning the encoding obtained when using f as the multi-dimensional basis function. The relations (2) and (19) then generalize to

$$\mathbf{c}\{f\} = \frac{1}{I} \sum_i w_i f(\mathbf{x}_i - \cdot) = \sum_{\mathbf{p}} \mathbf{K}\{f\}[\mathbf{p}, \cdot] * \mathbf{c}_{\text{mp}}[\mathbf{p}, \cdot] . \quad (21)$$

This compact notation which looks confusing at first glance, summarizes more or less everything said so far in a single equation. It is important to fully understand this equation before proceeding into the next section.

3.6 Derivatives

From (3) and (4), it is immediately clear that

$$\frac{d\mathbf{c}\{B\}[\tilde{\mathbf{x}}]}{db_u} = -\mathbf{c}\{B'_u\}[\tilde{\mathbf{x}}] \quad (22)$$

$$\frac{d\mathbf{c}\{B\}[\tilde{\mathbf{x}}]}{db_v} = -\mathbf{c}\{B'_v\}[\tilde{\mathbf{x}}] . \quad (23)$$

These can be computed by Alg. 1 using $\mathbf{K}\{B'_u\}$ and $\mathbf{K}\{B'_v\}$. The derivatives with respect to rotation and scale are more complicated. From (5), (6), we see that the four sums

$$\begin{aligned} I_1 &= \frac{1}{I} \sum_i u_i B'_u(\mathbf{x}_i - \tilde{\mathbf{x}}) & I_2 &= \frac{1}{I} \sum_i v_i B'_u(\mathbf{x}_i - \tilde{\mathbf{x}}) \\ I_3 &= \frac{1}{I} \sum_i u_i B'_v(\mathbf{x}_i - \tilde{\mathbf{x}}) & I_4 &= \frac{1}{I} \sum_i v_i B'_v(\mathbf{x}_i - \tilde{\mathbf{x}}) \end{aligned} \quad (24)$$

are needed. If B is here a separable polypuzzle of order n , so is uB'_u and vB'_v , while vB'_u and uB'_v are of order $n + 1$. This means that in order to compute

Algorithm 2 Computing CCFMs and derivatives through monopiece.

Compute $\mathbf{c}_{\text{mp}}[\mathbf{p}, \tilde{\mathbf{x}}]$ for each $\mathbf{p}$.

$\mathbf{c}\{B\} = \text{convert}(\mathbf{c}_{\text{mp}}, \mathbf{K}\{B\}, \mathbf{K}\{B\}, \mathbf{K}\{B\})$

$\mathbf{c}\{B'_u\} = \text{convert}(\mathbf{c}_{\text{mp}}, \mathbf{K}\{B'\}, \mathbf{K}\{B\}, \mathbf{K}\{B\})$

$\mathbf{c}\{B'_v\} = \text{convert}(\mathbf{c}_{\text{mp}}, \mathbf{K}\{B\}, \mathbf{K}\{B'\}, \mathbf{K}\{B\})$

$\mathbf{c}\{uB'_u\} = \text{convert}(\mathbf{c}_{\text{mp}}, \mathbf{K}\{xB'\}, \mathbf{K}\{B\}, \mathbf{K}\{B\})$

$\mathbf{c}\{uB'_v\} = \text{convert}(\mathbf{c}_{\text{mp}}, \mathbf{K}\{xB\}, \mathbf{K}\{B'\}, \mathbf{K}\{B\})$

$\mathbf{c}\{vB'_u\} = \text{convert}(\mathbf{c}_{\text{mp}}, \mathbf{K}\{B'\}, \mathbf{K}\{xB\}, \mathbf{K}\{B\})$

$\mathbf{c}\{vB'_v\} = \text{convert}(\mathbf{c}_{\text{mp}}, \mathbf{K}\{B\}, \mathbf{K}\{xB'\}, \mathbf{K}\{B\})$

Compute the desired derivatives using equations (22), (23), (28), (29).

the derivative with respect to image rotation we need more monopieces than are needed for the original encoding. This is a bit disappointing, but can be handled without too much extra overhead.

To see how to construct I_1, I_2, I_3, I_4 from a monopiece encoding, first rewrite I_1 as

$$I_1 = \frac{1}{I} \sum_i (u_i - \tilde{u}) B'_u(\mathbf{x}_i - \tilde{\mathbf{x}}) + \tilde{u} \frac{1}{I} \sum_i B'_u(\mathbf{x}_i - \tilde{\mathbf{x}}) = \quad (25)$$

$$= \mathbf{c}\{uB'_u\}[\tilde{\mathbf{x}}] + \tilde{u} \mathbf{c}\{B'_u\}[\tilde{\mathbf{x}}] . \quad (26)$$

By separating $uB'_u(\mathbf{x})$ into

$$uB'_u(\mathbf{x}) = uB'_1(u) \prod_{d \neq 1} B_d(\mathbf{x}[d]) , \quad (27)$$

$\mathbf{K}\{uB'_u\}$ can be separated according to (13) with $\mathbf{K}_1 = \mathbf{K}\{xB'_1\}$ and $\mathbf{K}_d = \mathbf{K}\{B\}$ for $d \neq 1$, and the encoding $\mathbf{c}\{uB'_u\}$ can be computed by Alg. 1.⁴ The other sums from (24) can be handled in a similar way, and altogether we get

$$\frac{d\mathbf{c}\{B\}[\tilde{\mathbf{x}}]}{d\alpha} = \mathbf{c}\{vB'_u\}[\tilde{\mathbf{x}}] + \tilde{v} \mathbf{c}\{B'_u\}[\tilde{\mathbf{x}}] - \mathbf{c}\{uB'_v\}[\tilde{\mathbf{x}}] - \tilde{u} \mathbf{c}\{B'_v\}[\tilde{\mathbf{x}}] , \quad (28)$$

$$\frac{d\mathbf{c}\{B\}[\tilde{\mathbf{x}}]}{ds} = - \mathbf{c}\{uB'_u\}[\tilde{\mathbf{x}}] - \tilde{u} \mathbf{c}\{B'_u\}[\tilde{\mathbf{x}}] - \mathbf{c}\{vB'_v\}[\tilde{\mathbf{x}}] - \tilde{v} \mathbf{c}\{B'_v\}[\tilde{\mathbf{x}}] - 2\mathbf{c}\{B\}[\tilde{\mathbf{x}}] . \quad (29)$$

Each of these $\mathbf{c}$ -terms can be computed from the monopiece encoding $\mathbf{c}_{\text{mp}}$ using Alg. 1 with the corresponding $\mathbf{K}$. To see how to construct things like $\mathbf{K}\{xB'\}$, we first note that the identity function $f(x) = x$ can be written

⁴ The symbol x is used to indicate that we are talking about a one-dimensional $\mathbf{K}$. To be strict, we should write $\mathbf{K}\{xB'\}$ as $\mathbf{K}\{x \mapsto xB'(x)\}$, not to be confused with $\mathbf{K}\{\mathbf{x} \mapsto uB'_u(\mathbf{x})\}$.

as a piecewise polynomial with $\mathbf{K}\{x\}[0, s] = s$ and $\mathbf{K}\{x\}[1, s] = 1$ for all s . This holds regardless of whether the shifts s are at the integers or at the 0.5-shifted integers. It is well-known that the product of two polynomials can be computed by a convolution of the polynomial coefficients, and for piecewise polynomials this convolution can be performed separately for each piece. In our notation this can be expressed compactly as

$$\mathbf{K}\{fg\}[\cdot, s] = \mathbf{K}\{f\}[\cdot, s] * \mathbf{K}\{g\}[\cdot, s] . \quad (30)$$

The complete algorithm for computing a channel-coded feature map together with its derivatives with respect to similarity transformation of the underlying image is given in Alg. 2. The function `convert`($\mathbf{c}_{\text{mp}}, \mathbf{K}_u, \mathbf{K}_v, \mathbf{K}_f$) mentioned in the pseudocode means running Alg. 1 for converting $\mathbf{c}_{\text{mp}}$ using $\mathbf{K}_u$ and $\mathbf{K}_v$ in the spatial dimensions and $\mathbf{K}_f$ in each feature dimension. For convenience, some standard coefficient matrices are given in Table 1, useful for creating first- and second-order B-spline CCFMs.

3.7 Required Monopieces

For a basis function of order k , all combinations of monopieces from order 0 to k are needed in each dimension, giving in total $(k + 1)^D$ monopieces. Furthermore, because of the terms from uB'_v and vB'_u , some monopieces of order $k + 1$ in one spatial dimension are needed – but only together with monopieces of order $k - 1$ in the other spatial dimension. This gives in total $\beta = (k + 1)^D + 2k(k + 1)^{(D-2)}$ monopieces. Some common cases are presented in Table 2.

4 Complexity Analysis

In this section, the computational complexity of the method is compared to a more direct method. In this analysis, we consider an image of size $M \times M$ from which $D - 2$ features are computed, such that the spatio-featural space has D dimensions like before. This feature map is encoded using N channels in each dimension, resulting in N^D channels in total. Let the basis function contain S pieces, such that exactly S channels are active at the same time in each dimension. This gives in total S^D active channels for each input pixel.

- **A) Direct Approach** The simplest thing to do is to loop through the image and accumulate the values of each of the five times S^D bins that are active for each pixel position. We need to compute the original encoding and the four derivatives separately. Each value of the encoding requires in the

order of D operations. Ignoring the index calculation for determining the active channels gives a total of $k_1 DS^D M^2$ operations, where k_1 is a constant depending on the computer architecture.

- **B) Piecewise Approach** As proposed in this paper, we start with computing a monopiece encoding from the image data using β monopieces, where β can be derived according to Sect. 3.7. Since each pixel is sent to β bins, this requires βM^2 monopiece evaluations, each requiring in average S operations. The entire monopiece encoding produces data of the size βN^D . From this representation, we create the final B-spline encoding and its derivatives by the technique described in Sect. 3.

Computing $\mathbf{c}\{B\}$ and each of the other 6 intermediate encodings in Alg. 2 requires βD one-dimensional convolutions with a convolution kernel of size S , each requiring SN^D operations. To combine the intermediate encodings into final derivatives requires only a single extra pass through the data (corresponding to (22), (23), (29), (28)), but this effort can be ignored. In total, this gives $k_2 S \beta M^2 + k_3 D \beta S N^D$ operations, where k_2 and k_3 are architecture dependent constant.

On the reference system (Intel Core Duo, 1.66 GHz), (dimensionless) integer constants were estimated from empirical break-even points as

$$k_1 = 17 \quad k_2 = 7 \quad k_3 = 19 \quad .$$

If the image is large compared to the number of channels, the complexity of the piecewise approach will be dominated by the first pass through the image, requiring $7S\beta M^2$ operations. In the limit of infinitely large images, the monopieces method is a factor $17DS^D/(7S\beta)$ faster than the direct method. As an example, consider the case of first-order B-spline basis functions ($S = 2$) and $D = 3$. Then $17DS^D = 408$ while $7S\beta = 168$ according to Table 2, i.e. the gain is about factor 2.4.

On the other hand, if the image size is small (e.g. 8×8), the influence of the number of channels increases and the piecewise approach becomes slower than the direct approach. For the example above and four channels in each dimension⁵, the direct method is faster by factor 5.4.

Furthermore, increasing the size S of the basis function while keeping everything else constant gives a dramatic advantage to the piecewise method, since the number of operations grows like S^D for the direct approach but at worst only linearly in $S\beta$ for the piecewise method.

⁵ It is not sensible to have more spatial channels than the original resolution, hence factor two is considered as the limiting case.

The previous considerations of the computational complexity can be used to predict the break-even point for the two different algorithms. Depending on the number of channels and the image size, the direct or the piecewise approach is preferable, c.f. Fig. 3. The empirical break-even points are plotted in the same figure and are basically consistent with the predictions. The slight overestimation for an intermediate number of channels might be explained by cache effects. The direct approach has larger memory requirements and therefore starts to use slower memory earlier than the piecewise approach does.

The quantitative time consumption and the empirical break-even point for a real system of course depend on the exact implementation and hardware used. Whereas the break-even points are rather stable, the time consumption can vary even on the same system, depending on the general load. Still, the absolute run-time is of interest when it comes to time-critical applications, e.g. real-time tracking. In Fig. 4, C++ implementations of the two approaches were compared in a number of situations. The experiments were run on a Pentium M running at 1.8 GHz. We conclude that the piecewise approach lifts the performance to video-realtime (25 Hz) in all cases, except for 2nd order, 8^3 channels, and larger images than 350×350 . Those cases where the direct approach is faster are not relevant concerning video-realtime, as both methods are significantly faster than 40ms anyway. This might be however different if several patches are to be tracked simultaneously. With the current implementation, two patches of size 128×128 can be tracked at 12.5 Hz using similarity transformations (changes of scale, rotation, and position) and 6 channels in each dimension, whereas the direct approach reaches 5 Hz. Finally, the piecewise method is also suitable for video-realtime encoding: PAL sequences can be encoded into 9×6 spatial channels and 16 feature channels at 25 Hz whereas the direct methods reaches only 10 Hz.

5 Discussion

In this paper, we have described one way of implementing channel-coded feature maps using separable piecewise polynomial basis functions. This approach shows favorable computational complexity compared to a direct encoding for images and image patches larger than about 20×20 pixels. The method can be applied to speed-up many feature extraction methods that are based on local histograms or statistics, e.g., SIFT features. The main advantage of the proposed method comes from the fact that much intermediate results can be reused in computing the derivatives. However, the amount of computation needed still grows rapidly when the basis functions become larger or the spatio-featural space higher-dimensional.

Fig. 3. Empirical and theoretical break-even points over image size (quadratic images) and number of channels (per dimension) using B-splines of first and second order. For the area above the curves, the piecewise method should be preferred.

Fig. 4. Empirical time consumption for two competing C++ implementations. Dashed: direct method. Solid: piecewise method.

This motivates trying to reduce the number of monopieces used. One motivation for channel-coded feature maps in the first place is to have a representation that responds smoothly to geometrical changes of the underlying image, with a coarse spatial resolution, but much information at each spatial position. Maybe these goals can be fulfilled with simpler basis functions, composed only of a subset of the monopieces needed for higher-order B-splines. This is related to the P-channel representation [3], where the number of monopieces used grows linearly in the number of feature dimensions. However, that representation is less smooth and less suited for taking derivatives. Finding a good trade-off between computational complexity and performance in any given application is subject to future research.

Acknowledgement

We would like to thank David Lowe for providing the original source code of SIFT.

References

- [1] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.
- [2] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(5):564 – 577, May 2003.
- [3] M. Felsberg and J. Hedborg. Real-time visual recognition of objects and scenes using p-channel matching. In *Proc. 15th Scandinavian Conference on Image Analysis*, volume 4522 of *LNCS*, pages 908–917, 2007.
- [4] Michael Felsberg and Gösta Granlund. P-channels: Robust multivariate m-estimation of large datasets. In *International Conference on Pattern Recognition*, Hong Kong, August 2006.
- [5] G. H. Granlund. An associative perception-action structure using a localized space variant information representation. In *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Kiel, Germany, September 2000.
- [6] E. Jonsson and M. Felsberg. Accurate interpolataion in appearance-based pose estimation. In *Proc. 15th Scandinavian Conference on Image Analysis*, 2007.
- [7] David G. Lowe. Object recognition from local scale-invariant features. In *IEEE Int. Conf. on Computer Vision*, Sept 1999.
- [8] Stephen Se, David Lowe, and Jim Little. Vision-based mobile robot lo-

- calization and mapping using scale-invariant features. In *Proc. Int. Conf. on Robotics and Automation*, 2001.
- [9] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR94)*, June 1994.
 - [10] Michael Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, November 1999.
 - [11] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, 2001.
 - [12] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In *Proc. Int. Conf. on Computer Vision*, volume 1, pages 212–219, October 2005.

	First order B-spline:	Second order B-spline:
$\mathbf{K}\{B\}$	$\begin{bmatrix} 1/2 & 1/2 \\ -1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1/8 & 3/4 & 1/8 \\ -1/2 & 0 & 1/2 \\ 1/2 & -1 & 1/2 \end{bmatrix}$
$\mathbf{K}\{B'\}$	$\begin{bmatrix} -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1/2 & 0 & 1/2 \\ 1 & -2 & 1 \end{bmatrix}$
$\mathbf{K}\{xB\}$	$\begin{bmatrix} 1/4 & -1/4 \\ 0 & 0 \\ -1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1/8 & 0 & -1/8 \\ -3/8 & 3/4 & -3/8 \\ 0 & 0 & 0 \\ 1/2 & -1 & 1/2 \end{bmatrix}$
$\mathbf{K}\{xB'\}$	$\begin{bmatrix} -1/2 & -1/2 \\ -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1/2 & 0 & -1/2 \\ 1/2 & 0 & -1/2 \\ 1 & -2 & 1 \end{bmatrix}$

Table 1

Some useful $\mathbf{K}$ -matrices. The topmost row in each matrix corresponds to polynomial order 0.

First-order B-spline, one feature f (12 monomials):									
1	u	u^2	v	uv	v^2	f	uf	u^2f	vf
							uvf	v^2f	
Second-order B-spline, one feature f (39 monomials):									
1	u	u^2	u^3	v	uv	u^2v	u^3v		
v^2	uv^2	u^2v^2	v^3	uv^3					
f	uf	u^2f	u^3f	vf	uvf	u^2vf	u^3vf		
v^2f	uv^2f	u^2v^2f	v^3f	uv^3f					
f^2	uf^2	u^2f^2	u^3f^2	vf^2	uvf^2	u^2vf^2	u^3vf^2		
v^2f^2	uv^2f^2	$u^2v^2f^2$	v^3f^2	uv^3f^2					

Table 2
Some examples of which monomials are required for different CCFMs.