Filtering with Gray-Code Kernels

Gil Ben-Artzi
Bar-Ilan University
Ramat Gan, Israel
gbenart@cs.biu.ac.il

Hagit Hel-Or
University of Haifa
Haifa, Israel
hagit@cs.haifa.ac.il

Yacov Hel-Or
The Interdisciplinary Center
Herzliya, Israel
toky@idc.ac.il

Abstract

In this paper we introduce a family of filter kernels - the Gray-Code Kernels (GCK) *and demonstrate their use in image analysis. Filtering an image with a sequence of Gray-Code Kernels is highly efficient and requires only* 2 operations per pixel *for each filter kernel, independent of the size or dimension of the kernel. We show that the family of kernels is large and includes the Walsh-Hadamard kernels amongst others. The GCK can also be used to approximate arbitrary kernels since a sequence of GCK can form a complete representation. The efficiency of computation using a sequence of GCK filters can be exploited for various real-time applications, such as, pattern detection, feature extraction, texture analysis, and more.*

1. Introduction

Convolving and correlating an image with filter kernels is one of the most important operations in image processing. This fundamental task is usually of high complexity, especially when an extensive set of large filter kernels are involved. A common approach for increasing efficiency is to design a set of specific kernels which are efficient to apply. The specific set can then be used to approximate any other kernel. Studies that took this course of action include the integral image [7], summed-area tables [3] and a generalized version of these called boxlets [5]. The main drawback of these approaches are that they allow only a very limited set of filter kernels to be computed using a fixed number of operations per pixel.

Our work is motivated by a previous study of Hel-Or [4] where a fast convolution scheme for the Walsh-Hadamard (WH) kernels was introduced for pattern detection. The computation cost of convolving each kernel is between 1 ops/pixel and up to at most $2logk$ ops/pixel for kernels of size $k \times k$.

In this paper, we introduce a family of filter kernels such that successive convolution of an image with a set of such filters is highly efficient and requires only *2 operations per pixel* for each filter kernel, regardless of the size or dimension of the filter. This family which we named *Gray-Code Kernels (GCK)* includes a very large set of filter kernels which can be used in a wide variety of applications, such as, pattern detection, feature extraction, texture analysis, and more.

2. The Gray-Code Kernels

Denote by $V_s^{(k)}$ a set of 1D kernels recursively expanded from an initial seed vector $\mathbf{s}$ as follows:

Definition 2.0.1

$$\begin{aligned} V_s^{(0)} &= \mathbf{s} \\ V_s^{(k)} &= \{[\mathbf{v}_s^{(k-1)} \;\; \alpha_k \mathbf{v}_s^{(k-1)}]\} \\ & \quad s.t. \;\; \mathbf{v}_s^{(k-1)} \in V_s^{(k-1)}, \alpha_k \in \{+1, -1\} \end{aligned}$$

where $\alpha_k \mathbf{v}^{(k)}$ denotes the multiplication of kernel $\mathbf{v}^{(k)}$ by the value α_k and $[\ldots]$ denotes concatenation.

The set of kernels and the recursive definition can be visualized as a binary tree of depth k. An example is shown in Figure 1 for $k = 3$. The nodes of the binary tree at level i represent the kernels of $V_s^{(i)}$. The leaves of the tree represent the 8 final kernels. The branches are marked with the values of α used to create the kernels.

Denote $|s| = t$ the length of s. It is easily shown that $V_s^{(k)}$ is an orthogonal set of 2^k kernels of length $2^k t$. Furthermore, given an orthogonal set of prefix vectors $\mathbf{s}_1, \ldots \mathbf{s}_r$, it can be shown that the union set $V_{s_1}^{(k)} \bigcup \ldots \bigcup V_{s_r}^{(k)}$ is orthogonal with $2^k r$ vectors of length $2^k t$. If $r = t$ the set forms a basis.

Figure 1 also depicts the fact that the values, $\alpha_1 \ldots \alpha_k$ along the tree branches, uniquely define a kernel in $V_s^{(k)}$.

Definition 2.0.2 *The sequence* $\boldsymbol{\alpha} = \alpha_1 \ldots \alpha_k$, $\alpha_i \in \{+1,-1\}$ *that uniquely defines kernel* $\mathbf{v} \in V_s^{(k)}$ *is the* **$\boldsymbol{\alpha}$-index** *of* $\mathbf{v}$.

We define the notion of **$\boldsymbol{\alpha}$-relation** between two filters:

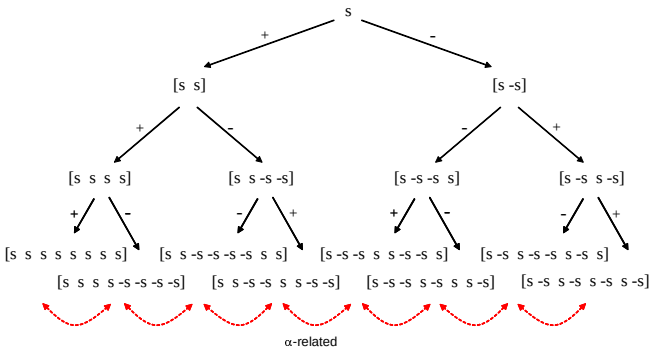

Figure 1. The set of GCK kernels and their recursive definition as a binary tree. Arrows indicate pairs of kernels that are α-related.

Definition 2.0.3 *Two kernels $\mathbf{v}_i, \mathbf{v}_j \in V_s^{(k)}$ are* **α-related** *iff the hamming distance of their* **α-index** *is one.*

The α-indices of two α-related kernels are $(\alpha_1 \dots \alpha_r, +1, \dots \alpha_k)$ and $(\alpha_1 \dots \alpha_r, -1, \dots \alpha_k)$. We denote the corresponding kernels as $\mathbf{v}_+$ and $\mathbf{v}_-$ respectively. Since $\alpha_1 \dots \alpha_r$ uniquely define a kernel in $V_s^{(r)}$, two α-related kernels always share the same prefix vector of length $2^r t = \Delta$. The arrows of Figure 1 indicate pairs of α-related kernels in the binary tree of depth $k = 3$. Note that not all possible pairs of kernels are α-related. Of special interest are sequences of kernels that are consecutively α-related.

Definition 2.0.4 *An ordered set of kernels $\mathbf{v}_0 \dots \mathbf{v}_r \in V_s^{(k)}$ that are consecutively α-related form a* **Gray Code Sequence (GCS)**. *The kernels are called* **Gray Code Kernels (GCK)**.

The term Gray Code relates to the fact that the series of α-indices associated with a Gray Code sequence forms a Gray Code [6]. An example of a Gray Code sequence is shown in Figure 1 where the kernels at the leaves of the tree are consecutively α-related. Note, however that this sequence is not unique and there are many possible ways of reordering the kernels to form a Gray Code Sequence.

Two α-related kernels share a special relationship which is the basis of this paper. It is described in the following:

Given two α-related kernels $\mathbf{v}_+, \mathbf{v}_- \in V_s^{(k)}$ their sum $\mathbf{v}_p$ and their difference $\mathbf{v}_m$ are defined as follows:

Definition 2.0.5

$$\begin{aligned} \mathbf{v}_p &= \mathbf{v}_+ + \mathbf{v}_- \\ \mathbf{v}_m &= \mathbf{v}_+ - \mathbf{v}_- \end{aligned}$$

Theorem 2.0.6 *Given two α-related kernels, $\mathbf{v}_+, \mathbf{v}_- \in V_s^{(k)}$ with a common prefix vector of length Δ, the following relation holds:*

$$[\mathbf{0}_\Delta \;\; \mathbf{v}_p] = [\mathbf{v}_m \;\; \mathbf{0}_\Delta]$$

where $\mathbf{0}_\Delta$ denotes a vector with Δ zeros.

Proof is given in [1]. For simplicity of explanation, we now expand $\mathbf{v} \in V_s^{(k)}$ to an infinite sequence such that $\mathbf{v}(i) = 0$ for $i < 0$ and for $i \geq 2^k t$. Using this convention, the relation $[\mathbf{0}_\Delta \;\; \mathbf{v}_p] = [\mathbf{v}_m \;\; \mathbf{0}_\Delta]$ can be rewritten in a new notation:

$$\mathbf{v}_p(i - \Delta) = \mathbf{v}_m(i)$$

With the new notation the above Theorem gives rise to the following Corollary:

Corollary 2.0.7

$$\begin{aligned} \mathbf{v}_+(i - \Delta) &= +\mathbf{v}_+(i) - \mathbf{v}_-(i) - \mathbf{v}_-(i - \Delta) \\ \mathbf{v}_-(i - \Delta) &= -\mathbf{v}_-(i) + \mathbf{v}_+(i) - \mathbf{v}_+(i - \Delta) \end{aligned}$$

Corollary 2.0.7 is the core principle behind the efficient filtering scheme introduced in this paper.

Let $\mathbf{b}_+$ and $\mathbf{b}_-$ be the signals resulting from convolving a signal with filter kernel $\mathbf{v}_+$ and $\mathbf{v}_-$ respectively. Then, by linearity of the filtering process (and Corollary 2.0.7) we have the following Lemma (proof is given in [1]):

Lemma 2.0.8

$$\begin{aligned} \mathbf{b}_+(i + \Delta_i) &= +\mathbf{b}_+(i) - \mathbf{b}_-(i) - \mathbf{b}_-(i + \Delta) \\ \mathbf{b}_-(i + \Delta_i) &= -\mathbf{b}_-(i) + \mathbf{b}_+(i) - \mathbf{b}_+(i + \Delta) \end{aligned}$$

This forms the basis of an efficient convolution scheme for projecting all signal windows onto all kernels. Given the result of convolving the signal with the filter kernel $\mathbf{v}_-$, the computation of convolving with the filter kernel $\mathbf{v}_+$ requires ***only*** 2 ***operations per pixel*** independent of the kernel size. Convolving with a Gray Code Sequence of kernels is denoted **Gray Code Filtering**.

2.1. Example - The 1D Walsh-Hadamard Kernels

Considering Definition 2.0.1, and setting the prefix string to $s = [1]$, we obtain that $V_s^{(k)}$ is the Walsh-Hadamard basis set of order 2^k. [1] A binary tree can be designed such that the Walsh-Hadamard kernels form a Gray Code sequence (i.e. are consecutively α-related) and are ordered in diadically increasing sequency order. Such a tree and a discussion of its efficiency in pattern detection is described in [4]. An example for $k = 2$ is shown in Figure 2 where every two consecutive kernels are α-related. For example, the first two kernels of order 4 are:

$$\begin{aligned} \mathbf{v_0} &= [1 \quad 1 \quad 1 \quad 1] \\ \mathbf{v_1} &= [1 \quad 1 \quad -1 \quad -1] \end{aligned}$$

They share the prefix string $[1 \quad 1]$, thus $\Delta = 2$. Their sum and difference are respectively $\mathbf{v}_p = [2 \quad 2 \quad 0 \quad 0]$ and $\mathbf{v}_m = [0 \quad 0 \quad 2 \quad 2]$ and Theorem 2.0.6 holds with:

$$\mathbf{v}_p(i - 2) = \mathbf{v}_m(i)$$

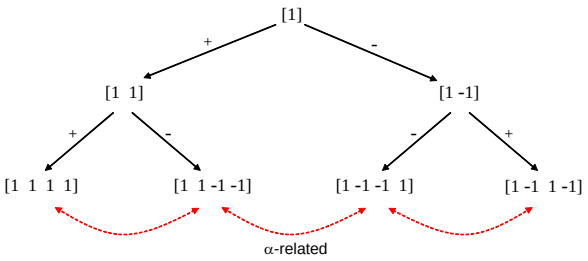

Figure 2. Binary tree for the Walsh-Hadamard basis set of order 4. Consecutive kernels are α-related, as shown by the arrows.

Thus, according to Corollary 2.0.7 the first two Walsh-Hadamard kernels and their shifted versions are related by:

$$\mathbf{v}_1(i-2) \quad = \quad -\mathbf{v}_1(i) + \mathbf{v}_0(i) - \mathbf{v}_0(i-2)$$

Thus, given the filtering result with the first Walsh-Hadamard kernel, the filtering with the second kernel requires only 2 operations (additions/subtractions) per pixel.

Ordering the Walsh-Hadamard kernels to form a Gray-Code Sequence, the windowed Walsh-Hadamard transform can be performed on a signal using only 2 operations per pixel per kernel regardless of signal and kernel size.

3. Generalized Gray Code Kernels

The previous sections can be generalized to form GCKs in any dimension and for various sizes other than 2^k. In addition, the α-index can be extended to include any scalar, allowing a larger set of possible GCK, at an additional cost of ops/pixel. The basis for these generalizations can be found in [1]. The core principle is to form a new set of higher dimensional GCKs by using cross products of 1D GCKs. Two high dimensional filters are then α-related if one of their separable 1D components are α-related. For example consider the 2D kernels:

$$W_1 = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{bmatrix} \quad W_2 = \begin{bmatrix} 2 & 2 & -2 & -2 \\ 3 & 3 & -3 & -3 \end{bmatrix}$$

These kernels are separable:

$$\begin{matrix} W_1: \\ W_2: \end{matrix} \quad \mathbf{v}_F = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad \times \quad \begin{matrix} \mathbf{v}_1 = [1 & 1 & 1 & 1] \\ \mathbf{v}_2 = [1 & 1 & -1 & -1] \end{matrix}$$

Such that their 1D components v_1, v_2 are α-related.

Given the filtering result B_1 of the 2D image I with kernel W_1, the filtering B_2 with kernel W_2 can be calculated using only 2 operations per pixel:

$$B_2(q_1, q_2+2) = -B_2(q_1, q_2) + B_1(q_1, q_2) - B_1(q_1, q_2+2)$$

Details of this extension can be found in [2].

[1]For simplicity, we use non-unitary basis vectors. Normalization is straightforward and does not affect the defi nitions and results in this paper.

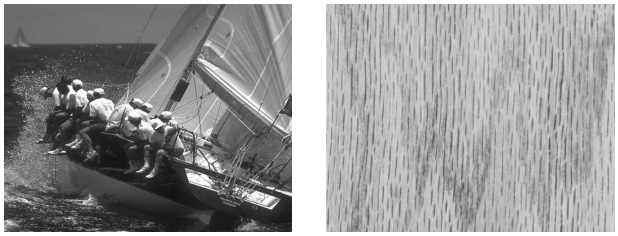

Figure 3. Natural and Texture images used in experiments.

4. Experiments

The most attractive property of the Gray-Code Kernel (GCK) framework is that it allows successive filtering using only 2 ops/pixel per filter kernel. However, the sequence of filters must be ordered according to their α-indices to form one of the many possible Gray-Code Sequences. This section tests the implications of this requirement in a Pattern-Matching process using the rejection framework proposed by Hel-Or [4].

Consider the Euclidean distance $d_E^2(\mathbf{p}, \mathbf{w})$ between a pattern $\mathbf{p}$ and an image window $\mathbf{w}$ resulting by projecting $\mathbf{p}$ and $\mathbf{w}$ onto a set of orthogonal kernels $\{\mathbf{v}_1 \dots \mathbf{v}_r\}$: $d_E^2(\mathbf{p}, \mathbf{w}) = \sum_{i=1}^{r} \left(\mathbf{v}_i^T \mathbf{p} - \mathbf{v}_i^T \mathbf{w}\right)^2$. Using the rejection framework, we project the pattern and each image window onto all r filter kernels consecutively, where in each projection stage j, a lower bound on the Euclidean distance is formed by the partial sum $LB_j = \sum_{i=1}^{j} \left(\mathbf{v}_i^T \mathbf{p} - \mathbf{v}_i^T \mathbf{w}\right)^2$. At each stage, image windows whose lower bound is above a predefined threshold are rejected. As the number of projections j increases, the lower bound becomes tighter and in turn more image windows are rejected. From earlier experimental results it was found that when the LB reaches 80% of the actual window-pattern distance, almost all non-matching windows are rejected [4]. Consider the contribution $c(\mathbf{v}_i)$ of filter kernel $\mathbf{v}_i$ to the Euclidean distance: $c(\mathbf{v}_i) = \frac{\left(\mathbf{v}_i^T \mathbf{p} - \mathbf{v}_i^T \mathbf{w}\right)^2}{d_E(\mathbf{p}, \mathbf{w})}$ averaged over all image windows. The larger $c(\mathbf{v}_i)$, the more it contributes, on average, to the Lower bound. Thus, in order to reach the 80% goal as fast as possible, it is advantageous to first project onto the filter kernels whose c values are large. This implies that the projection order of kernels determines the total number of kernels required to reach the 80% goal, and consequently, the total number of ops/pixel to complete the process.

We used the Walsh-Hadamard kernels for testing and comparing the efficiency of three computation schemes. Each computation scheme uses a different sequence of kernels and as a consequence, results in a different operational cost per kernel. The three computation schemes are defined by the order of kernels:

	N-N +DC	N-N -DC	T-T +DC	T-T -DC	T-N +DC	T-N -DC	N-T +DC	N-T -DC
Opt.	7	85	17	50	9	116	1	39
Seq.	7	88	131	196	11	162	1	139
GCK	7	88	18	58	10	125	1	42

Table 1. The number of kernels dictated by the order of each computation scheme. Each experiment is denoted by the pattern-image pair (N=Natural, T=Texture) and whether the DC was included (+DC) or not (-DC).

- **Optimal** - The best order of kernels dictated by their contribution c. However, in terms of computation, since it does not form a Gray Code Sequence, projecting onto each kernel, naively requires $2logk$ ops/pixel.
- **Sequency** - The order of increasing spatial frequency of each kernel. The computation cost requires from 1 ops/pixel up to $2logk$ ops/pixel, exploiting the tree structure of Figure 2 [4]. This computation scheme is known to best preform for natural images.
- **GCS** - The order of kernels that form a GCS and is as similar as possible to the Optimal order (i.e. contains as few additional kernels as possible beyond those of the Optimal sequence). Using the filtering scheme described above, filtering with each kernel can be performed using only 2 ops/pixel.

The results using the 3 computation schemes were compared over 4 pattern-image pairs. Two images of size 512×512 were chosen (see Figure 3), representing a 'natural' and a 'texture' image. A 32×32 window was chosen randomly from each image and these served as the patterns, denoted as 'natural pattern' and 'texture pattern'. Each case of pattern and image pair was tested both with and without the DC kernel. Table 1 presents the number of kernel required by each computation scheme in order to reach the 80% goal. This number is dictated by the constraints on the order which characterized each computation scheme. Figure 4 shows the total number of ops/pixel required by the 3 computation schemes for the 4 pattern-image cases with and without the DC value. It can be seen that the GCS scheme always required fewer ops/pixel than the other two computation schemes. Details of the experimental procedure, and analysis can be found in [1]).

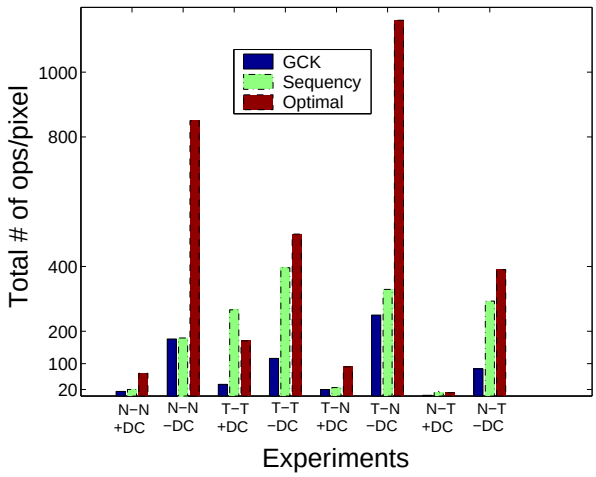

Figure 4. The total number of ops/pixel required by each of the computation schemes. Each experiment is denoted as in Figure 1.

5. Conclusions

The Gray Code Kernels introduce the possibility of computing the projections of all image windows onto a sequence of kernels using only *2 operations* per pixel per kernel. Moreover, the computation cost is independent of the kernel size and dimension. In addition, for integer seed vectors, the computation is performed using only integer operations which are much faster than floating operations in most hardware. We also show that using different seed kernels $\mathbf{s}$, a large family of GCKs can be created. Using various orders of the kernels numerous Gray Code Sequences can be formed.

The unique properties of the GCK framework make it an attractive choice for many applications that require efficient applications of filter banks, such as feature extraction, segmentation, block matching, texture analysis and synthesis.

References

[1] G. Ben-Artzi. The gray-code filter kernels (gck). Master's thesis, Bar-Ilan University, Ramat-Gan, Israel, 2003.

[2] G. Ben-Artzi, Y. Hel-Or, and H. Hel-Or. Generalized gray-code filter kernels (gcf). In preparation.

[3] F. C. Crow. Summed-area tables for texture mapping. In *Proc. SIGGRAPH*, volume 18, pages 207–212, 1984.

[4] Y. Hel-Or and H. Hel-Or. Real time pattern matching using projection kernels. In *Proc. ICCV*, pages 1486–1493, Nice, France, 2003.

[5] P. Simard, L. Bottou, P. Haffner, and Y. LeCun. Boxlets: a fast convolution algorithm for neural networks and signal processing. In *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[6] S. Skiena. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley, Reading, Massachusetts, 1990.

[7] P. Viola and M. Jones. Robust real-time object detection. Technical Report CRL 2001/01, The Cambridge Research Laboratory, Febuary 2001.