The Gray Code Kernels

Gil Ben-Artzi Hagit Hel-Or Yacov Hel-Or

Bar-Ilan University Haifa University IDC

Motivation

- Image filtering with a successive set of kernels is very common in many applications:
  - Pattern classification
  - Pattern matching
  - Texture analysis
  - Image Denoising

In some applications applying a large set of filter kernels is prohibited due to time limitation.

Example 1: Pattern detection

- **Pattern Detection**: Given a pattern subjected to some type of deformations, detect occurrences of this pattern in an image.

- Detection should be:
  - Accurate (small number of mis-detections/false-alarms).
  - As fast as possible.

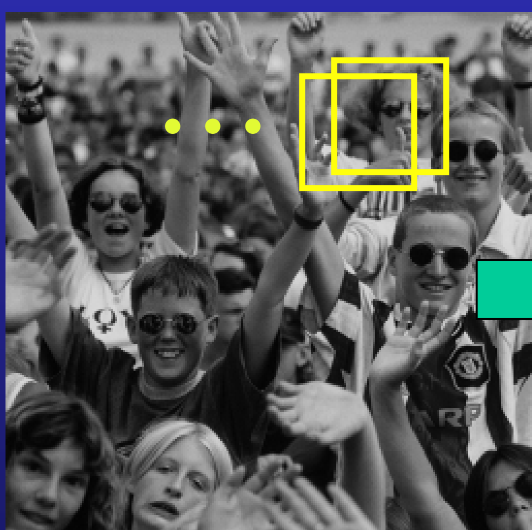
Pattern Detection as a Classification Problem

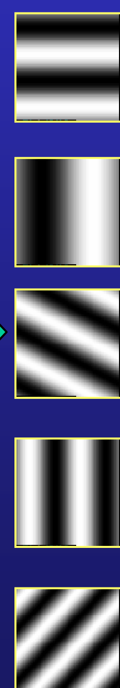
Pattern detection requires a separation between two classes:

a. The Target class.

b. The Clutter class.

The detection complexity is dominated by the feature extraction





z_1

z_2

z_3

z_4

z_5

Classifier

$$C\{z_1, z_2, z_3, \cdots\}: \; \Re^n \to \{+1, -1\}$$

Feature extraction

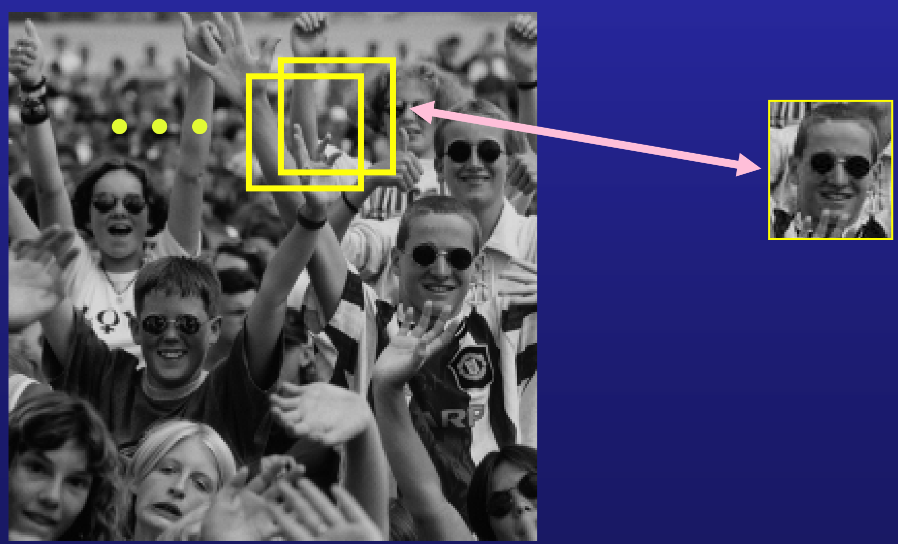
Feature Selection

- In order to optimize classification complexity, the feature set should be selected according to the following criteria:

1. Informative: high “separation” power
2. Fast to apply.

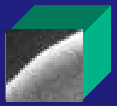
Example 2: Pattern Matching

- A known pattern is sought in an image.
- The pattern may appear at any location in the image.
- A degenerated classification problem.



The Euclidean Distance

$$d_E(u,v)=\sum_{x,y\in N}\left[I(x-u,y-v)-P(x,y)\right]^2$$



$$d_E(u,v,t)=\sum_{x,y,t\in N}\left[I(x-u,y-v,t-w)-P(x,y,t)\right]^2$$

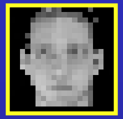
Complexity (2D case)

	Average # Operations per Pixel	Space	Integer Arithm.	Run Time for 1Kx1K Image 32x32 pattern PIII, 1.8 Ghz
Naive	+: $2k^2$ *: k^2	n^2	Yes	5.14 seconds
Fourier	+: $36 \log n$ *: $24 \log n$	n^2	No	4.3 seconds

Far from real-time performance

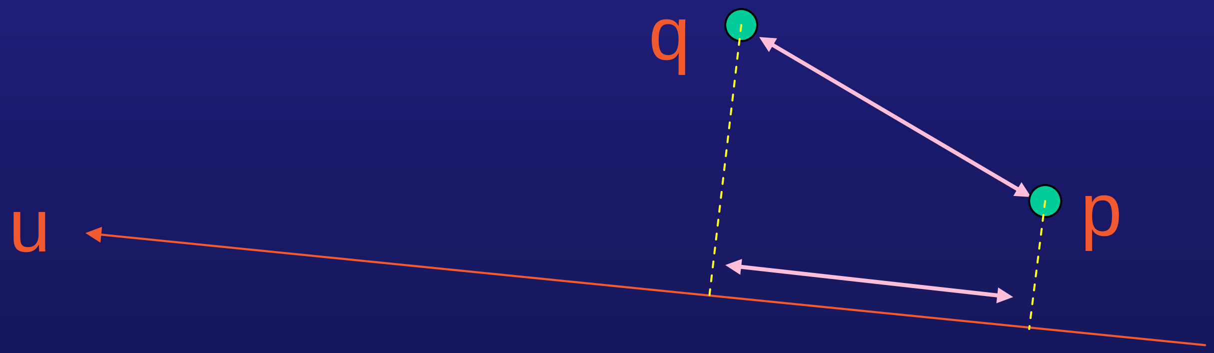
Suggested Solution: Bound Distances using Projection Kernels (Hel-Or[2] 03)

- Representing an image window and the pattern as points in R^{kxk}:

$d_E(p,q)= ||p-q||^2= ||$  -  $||^2$

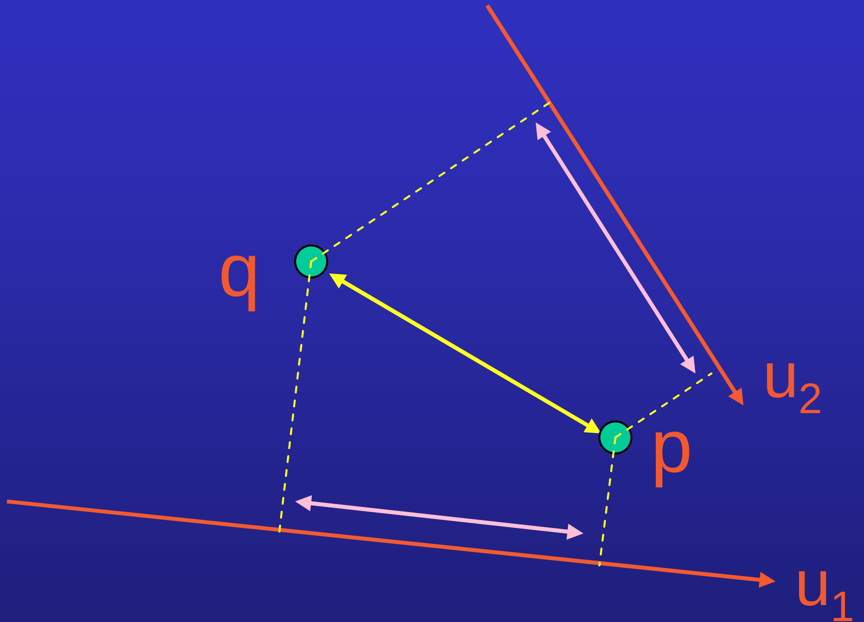
- If p and q were projected onto a kernel u, it follows from the Cauchy-Schwarz Inequality:

$$d_E(p,q) \geq |u|^{-2}\, d_E(p^T u,\, q^T u)$$



Distance Measure in Sub-space (Cont.)

- If *q* and *p* were projected onto a set of kernels *[U]:*

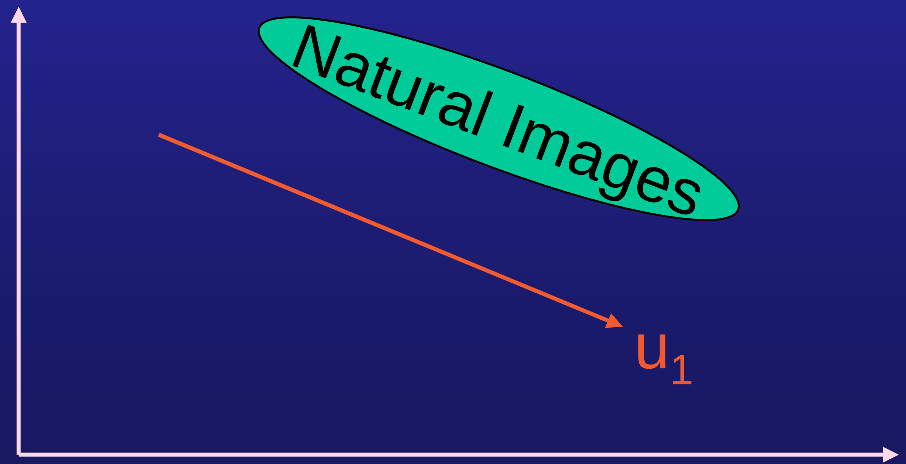


$$d_E(p,q) \geq \sum_{k=1}^{r} \frac{1}{S_k^2} d_E\left(p^T u_k, q^T u_k\right)$$

How can we Expedite the Distance Calculations?

Two necessary requirements:

1. Choose informative projecting kernels *[U];* having high probability to be parallel to the vector *p-q*.
2. Choose projecting kernels that are fast to apply.



Our Goal

Design a set of filter kernels with the following properties:

– "Informative" in some sense.
– Efficient to apply successively to images.
– Consists of a large variety of kernels.
– Forms a basis, thus allowing approximating <u>any</u> set of filter kernels.

Fast Filter Kernels

- Previous work:
  - Summed-area table / Franklin [1984]
  - Boxlets/ Simard, et. Al. [1999]
  - Integral image/ Viola & Jones [2001]

Average / difference kernels

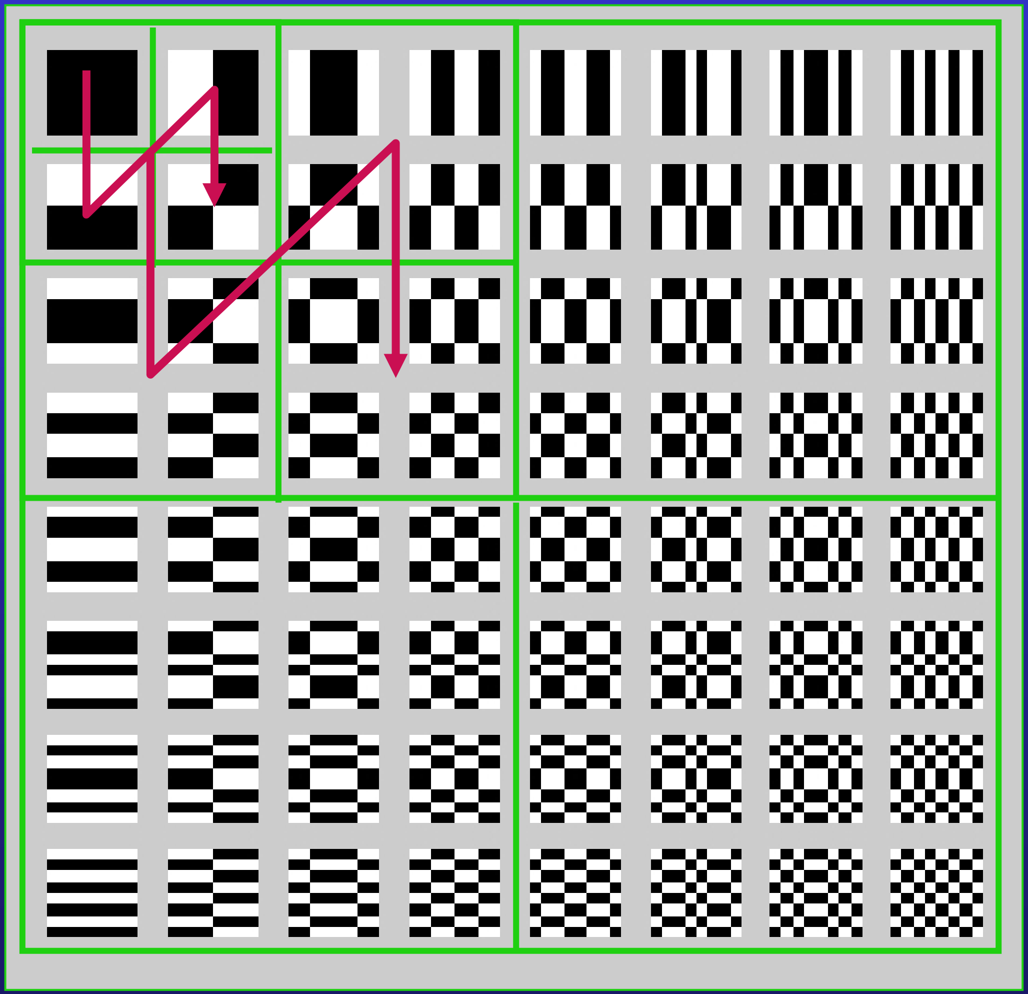
- Limitations:
  - A limited variety of filter kernels.
  - Approximation of large sets might be inefficient.
  - Does not form a basis and thus inefficient to compose other kernels.

Our work based upon

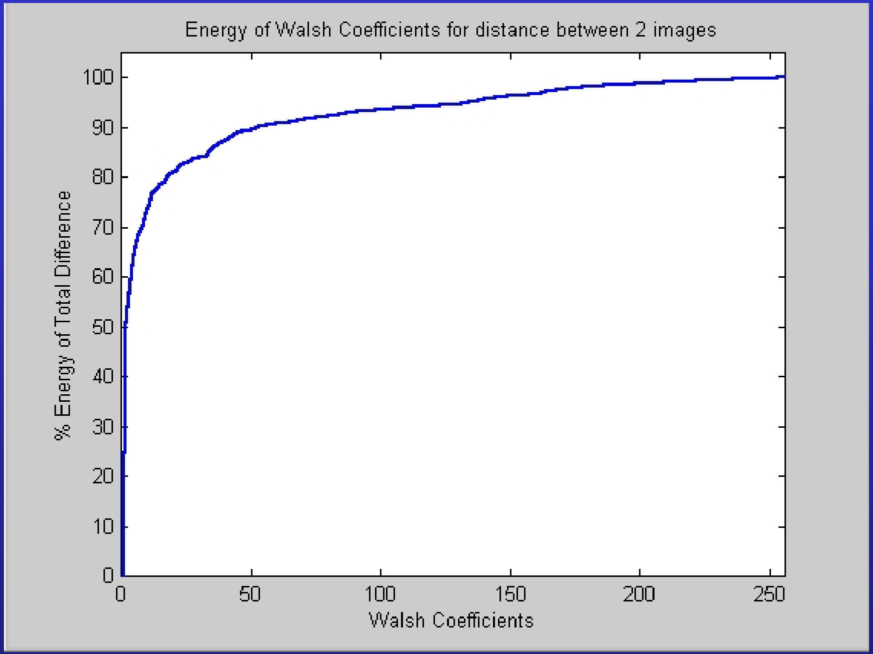
Real-Time projection kernels [Hel-Or^2 03]

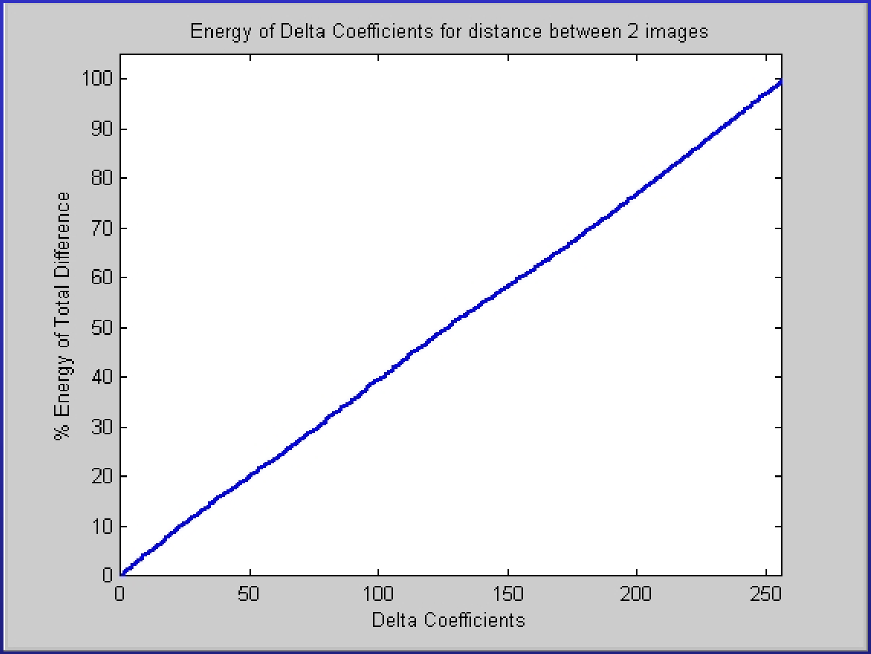
- A set of Walsh-Hadamard basis kernels.
- Each window in a natural image is closely spanned by the first few kernel vectors.
- Can be applied very fast in a recursive manner.

The Walsh-Hadamard Kernels:



Walsh-Hadamard v.s. Standard Basis:

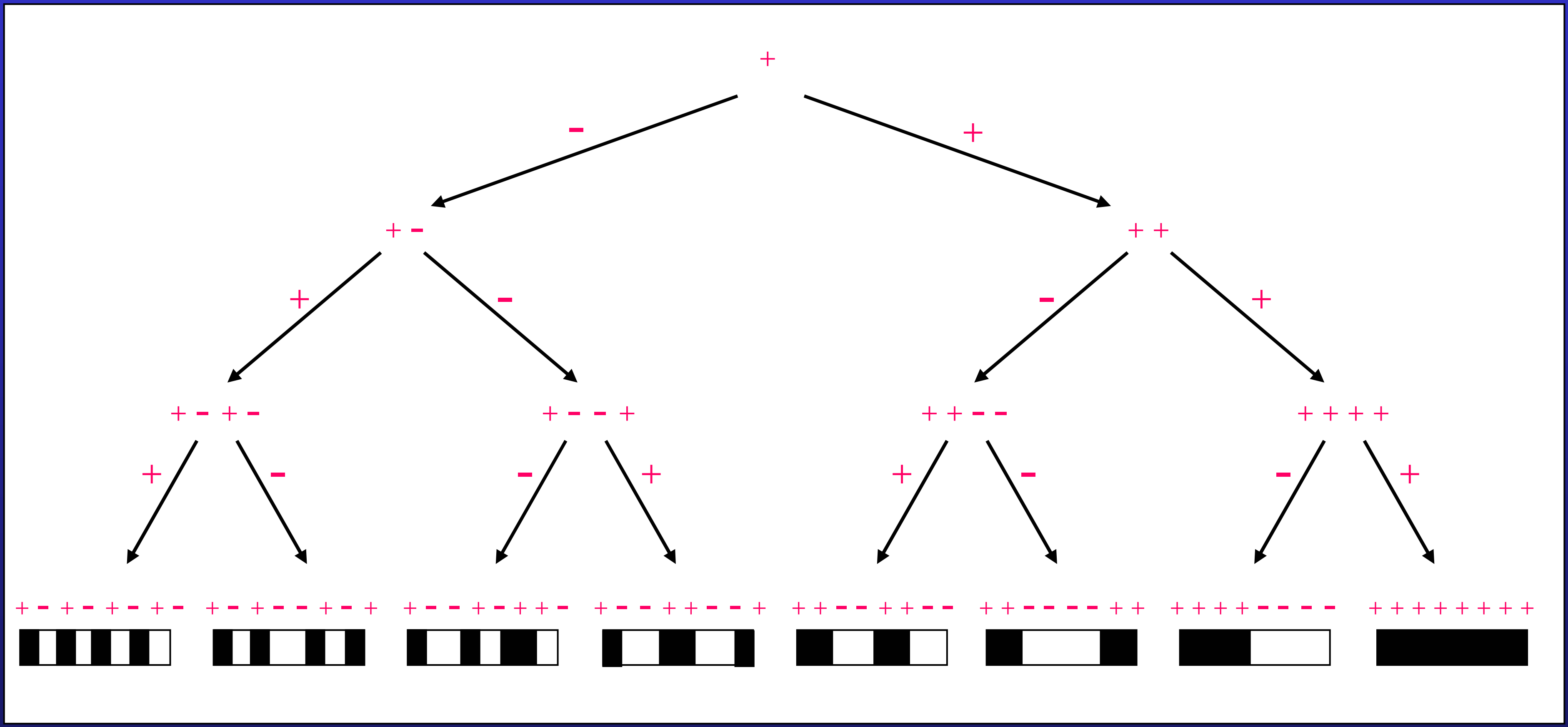




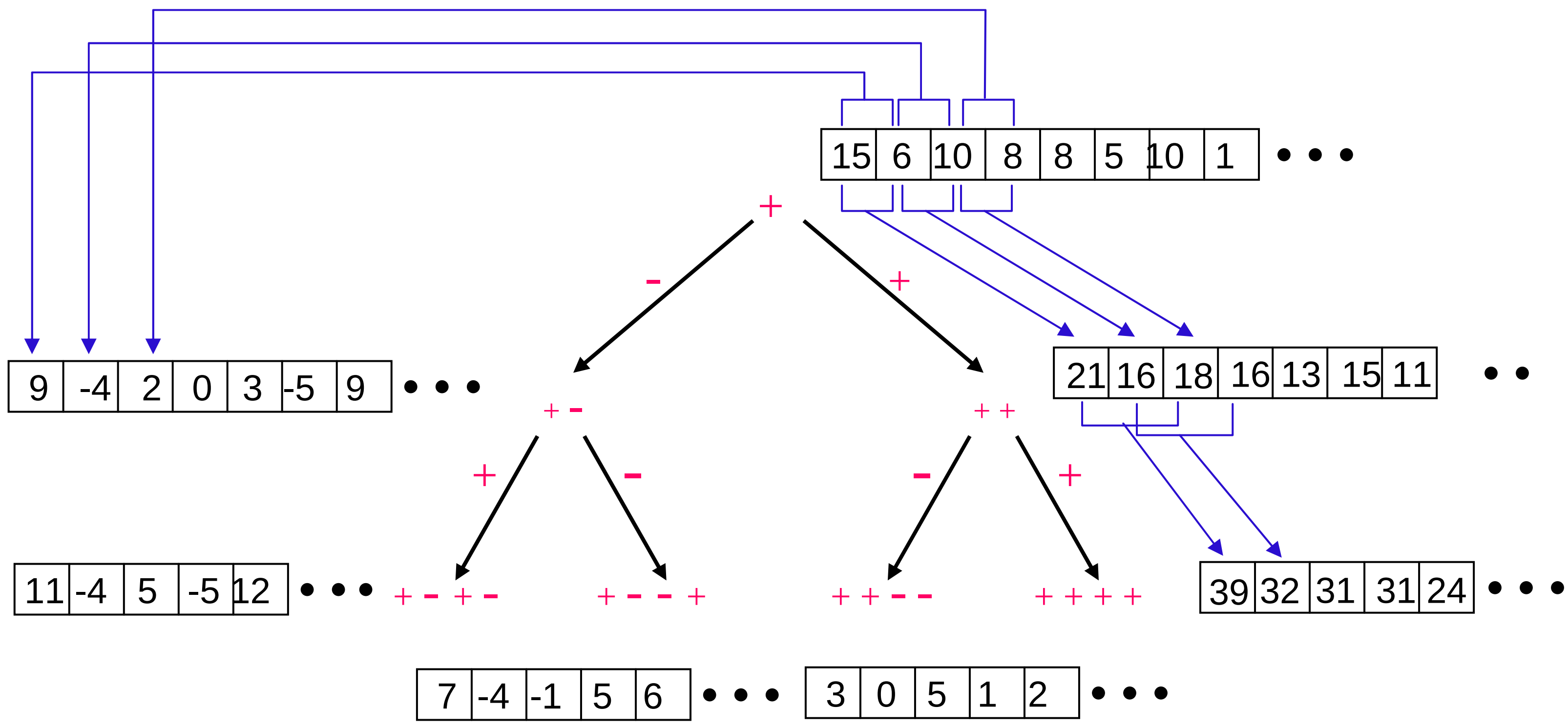
The lower bound for distance value in % v.s. number of Walsh-Hadamard projections,
Averaged over 100 pattern-image pairs of size 256x256 .

The lower bound for distance value in % v.s. number of standard basis projections,
Averaged over 100 pattern-image pairs of size 256x256 .

The Walsh-Hadamard Tree (1D case)

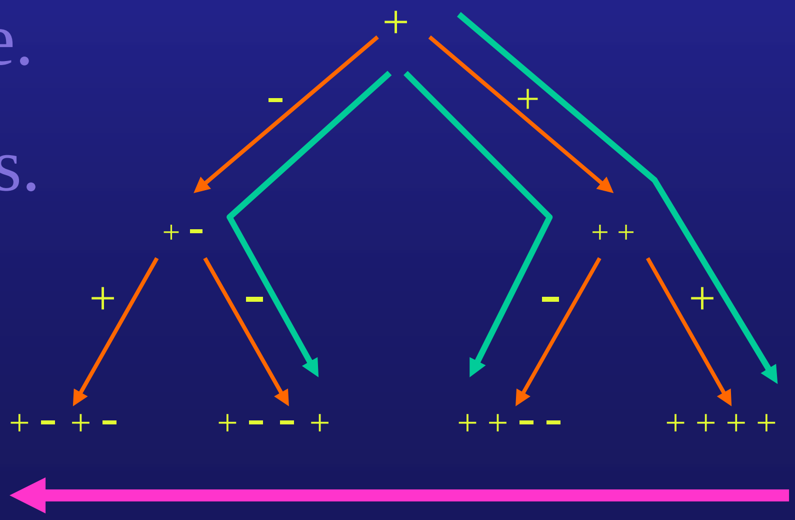


The Walsh-Hadamard Tree - Example



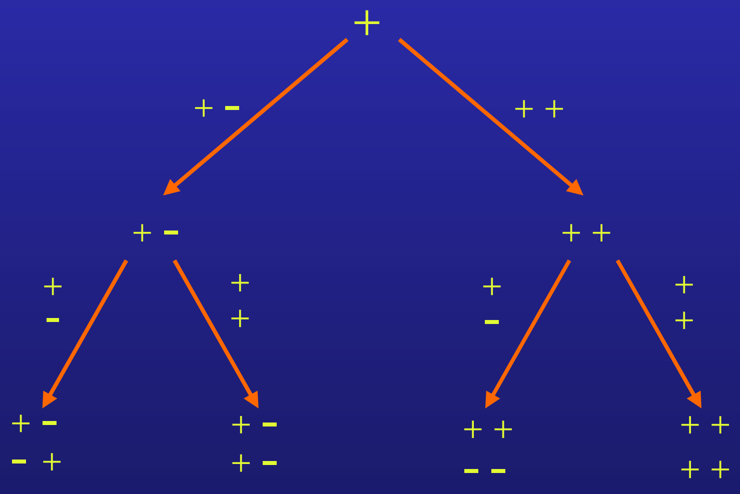
Properties:

- Descending from a node to its child requires one addition operation per pixel.
- The depth of the tree is *log k* where k is the kernel's size.
- Successive application of WH kernels requires between O(1) to O(log k) ops per kernel per pixel.
- Requires *n log k* memory size.
- Linear scanning of tree leaves.



Walsh-Hadamard Tree (2D):

- For the 2D case, the projection is performed in a similar manner where the tree depth is *2log k*
- The complexity is calculated accordingly.

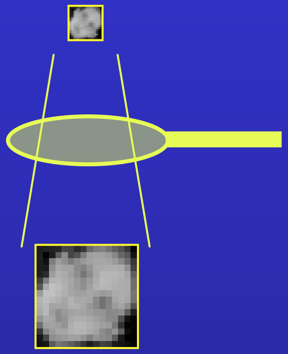


Construction tree for 2x2 basis

WH for Pattern Matching

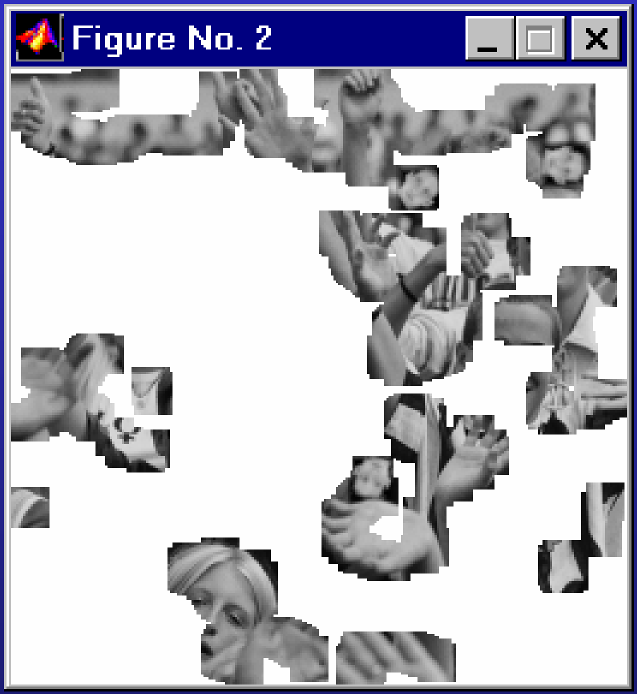
- Iteratively apply Walsh-Hadamard kernels to each window w_i in the image.
- At each iteration and for each w_i calculate a lower-bound Lb_i for $|p-w_i|^2$.
- If the lower-bound Lb_i is greater than a pre-defined threshold, reject the window w_i and ignore it in further projections.

Example:

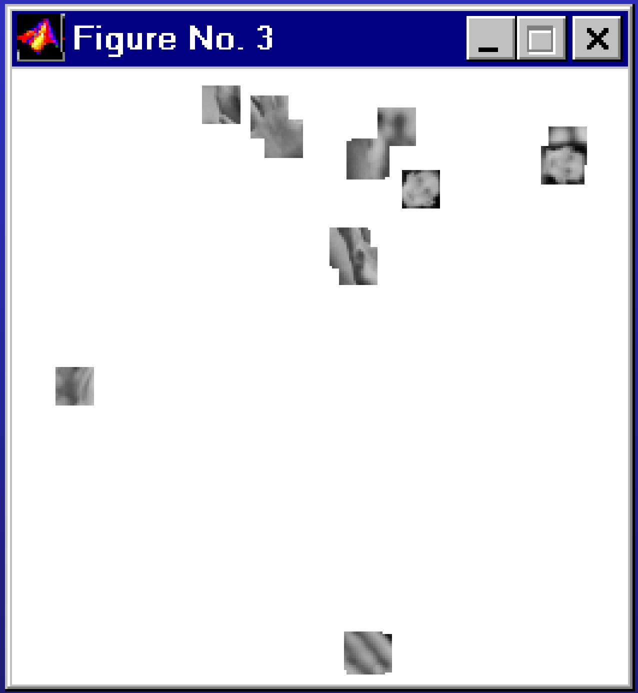


Sought Pattern

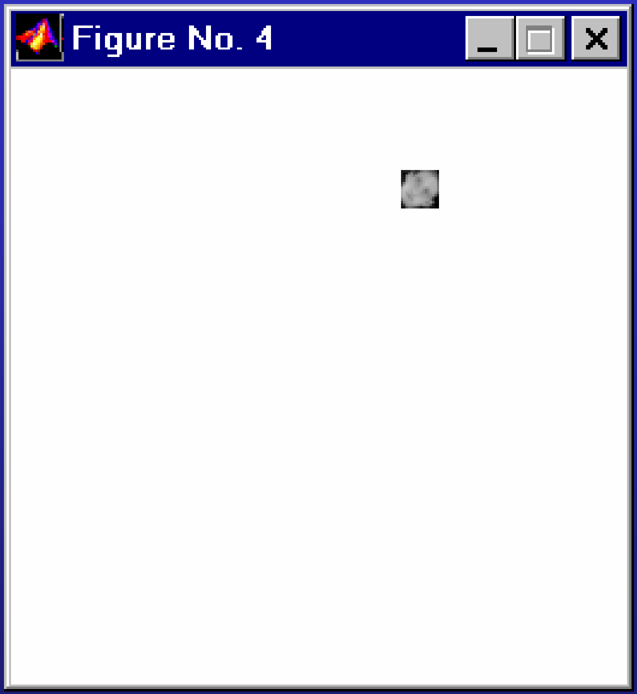
Initial Image: 65536 candidates



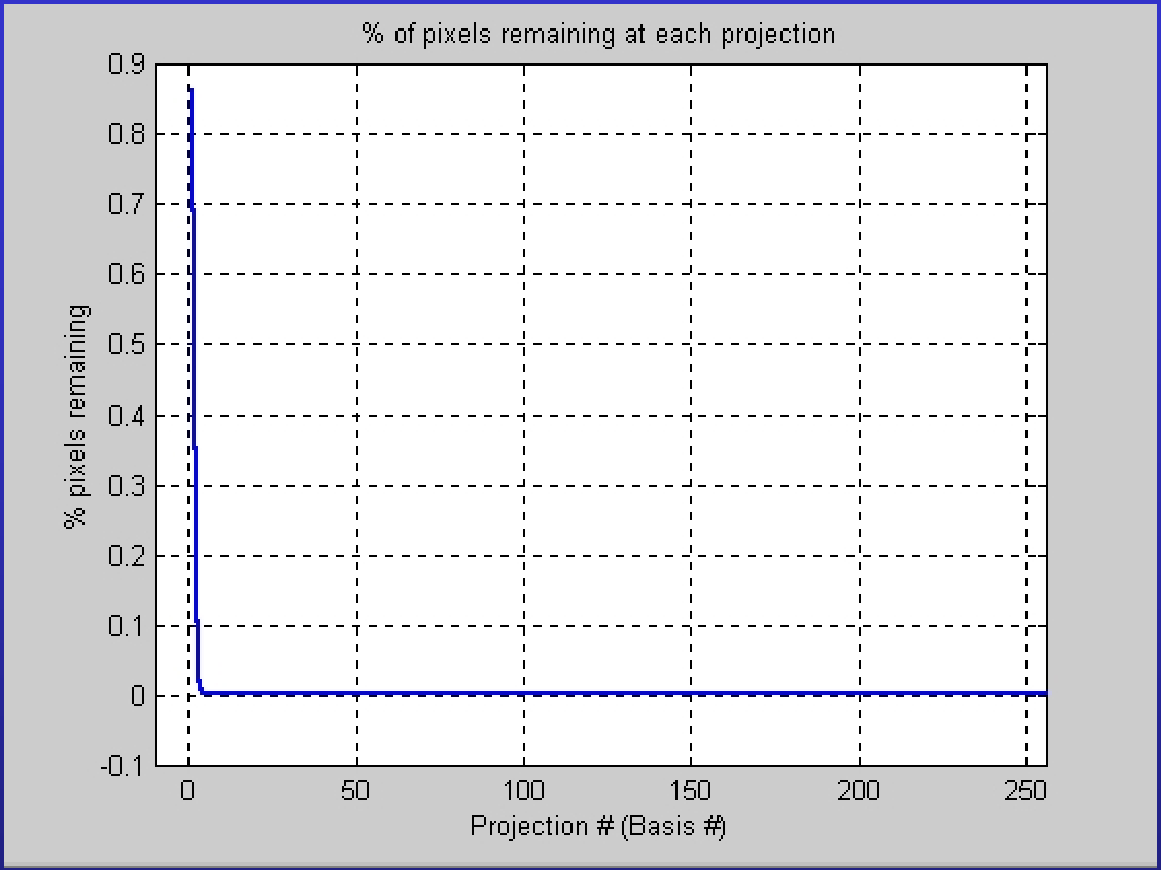
After the 1^{st} projection: 563 candidates



After the 2nd projection: 16 candidates



After the 3rd projection: 1 candidate



Percentage of windows remaining following each projection,
averaged over 100 pattern-image pairs.

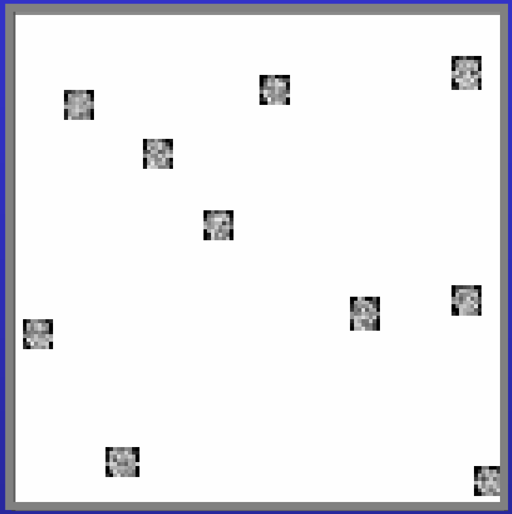
Image size = 256x256, pattern size = 16x16.

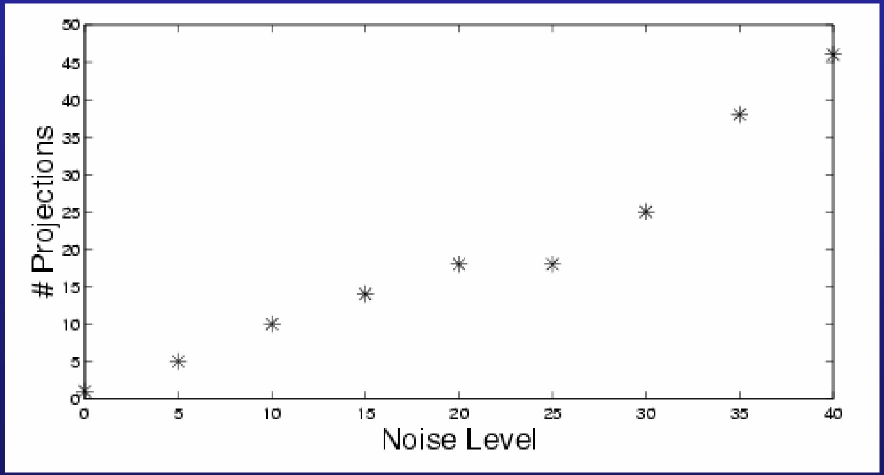
Example with Noise

Original

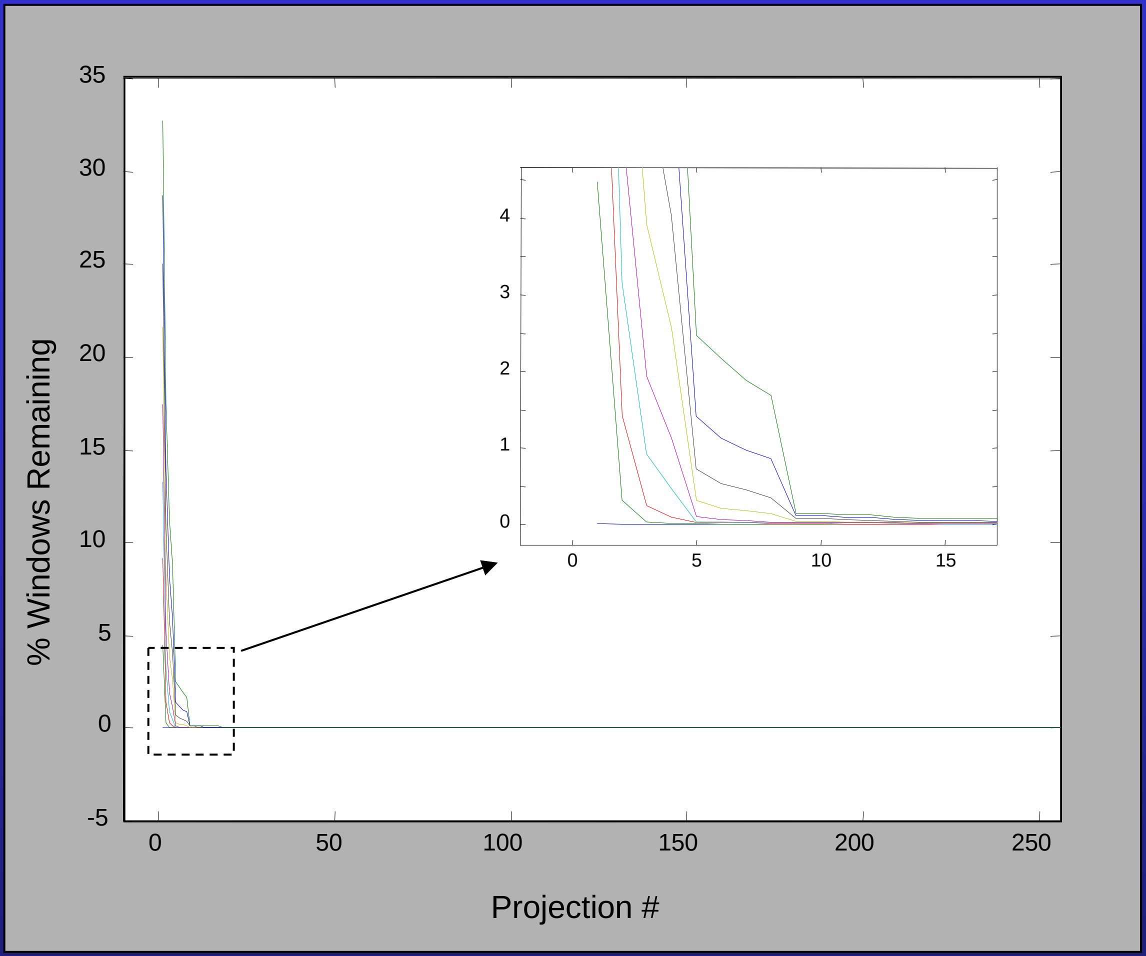
Noise Level = 40

Detected patterns.





Number of projections required to find all patterns, as a function of noise level. (Threshold is set to minimum).



Percentage of windows remaining following each projection,
at various noise levels.

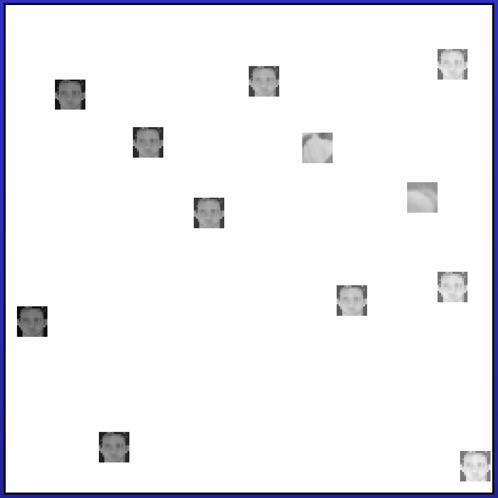
Image size = 256x256, pattern size = 16x16.

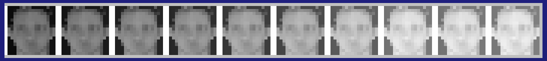
DC-invariant Pattern Matching

Original

Illumination gradient added

Detected patterns.





Five projections are required to find all 10 patterns (Threshold is set to minimum).

Complexity (2D case)

	Average # Operations per Pixel	Space	Integer Arithm.	Run Time for 1Kx1K Image 32x32 pattern PIII, 1.8 Ghz
Naive	+: $2k^2$ *: k^2	n^2	Yes	4.86 seconds
Fourier	+: $36 \log n$ *: $24 \log n$	n^2	No	3.5 seconds
New	+: $2 \log k + \varepsilon$	$n^2 \log k$	Yes	78 msec

Advantages:

- WH kernels can be applied very fast.
- Projections are performed with additions/subtractions only (no multiplications).
- Integer operations (3 times faster for additions).
- Possible to perform pattern matching at video rate.
- Can be easily extended to higher dim.

Limitations

– Limited set - only the Walsh-Hadamard kernels.

– Each kernel is applied in O(1)-O(d log k)

– Limited order of kernels.

– Limited to dyadic sized kernels.

– Requires maintaining d log k images in memory.

The Gray Code Kernels (GCK):

- Allowing convolution of large set of kernels in O(1):
  - Independent of the kernel size.
  - Independent of the kernel dimension.
  - Allows various computation orders of kernels.
  - Various size of kernels other than 2^n.
  - Requires maintaining 2 images in memory.

The Gray Code Kernels – Definitions (1D)

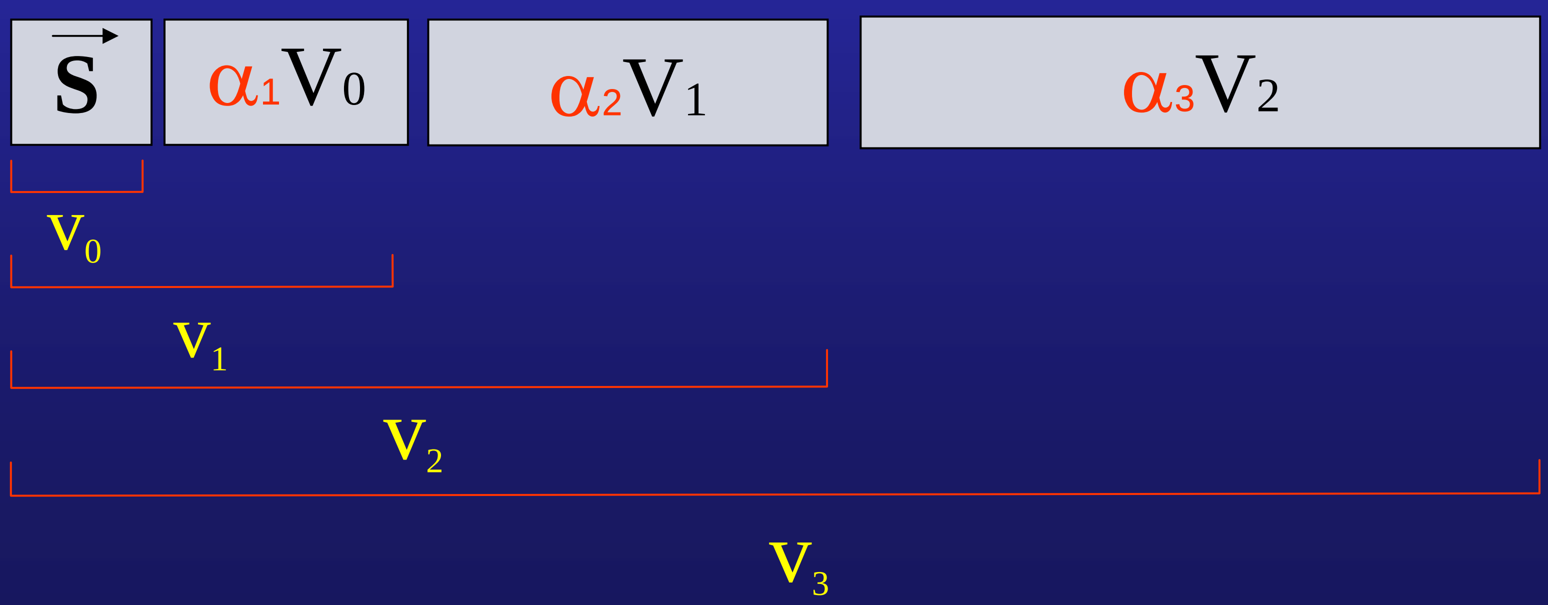
Input

1. A seed vector $\vec{s}$.

2. A set of coefficients $\alpha_1, \alpha_2 \ldots \alpha_k \in \{+1,-1\}$.

Output

A set of recursively built kernels :



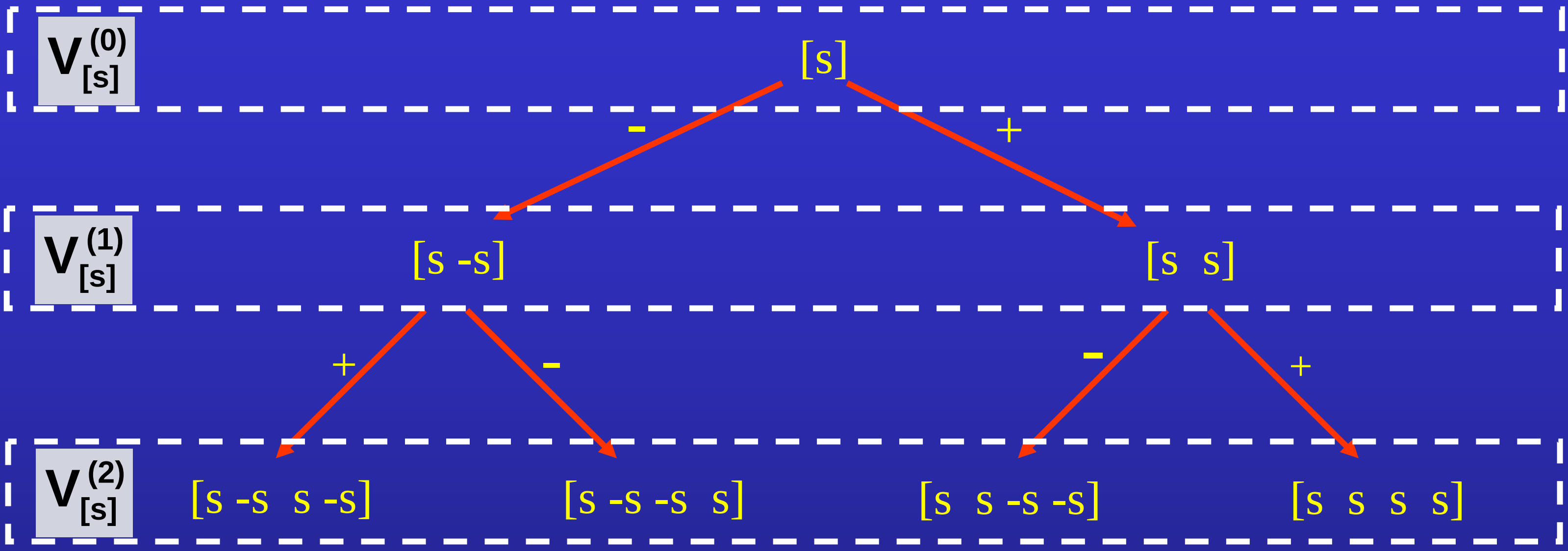
GCK - Formal Definitions

$$\mathrm{V}_{\mathrm{s}}^{(0)} = \vec{s}$$

$$\mathrm{V}_{\mathrm{s}}^{(\mathrm{k})} = \left\{ \left[\vec{v}^{(k-1)} \quad \alpha_k \vec{v}^{(k-1)} \right] \right\}$$

$$s.t. \quad \vec{v}^{(k-1)} \in \mathrm{V}_{\mathrm{s}}^{(\mathrm{k}\text{-}1)} \quad and \quad \alpha_k \in \{+1,-1\}$$

1 Dim GCK

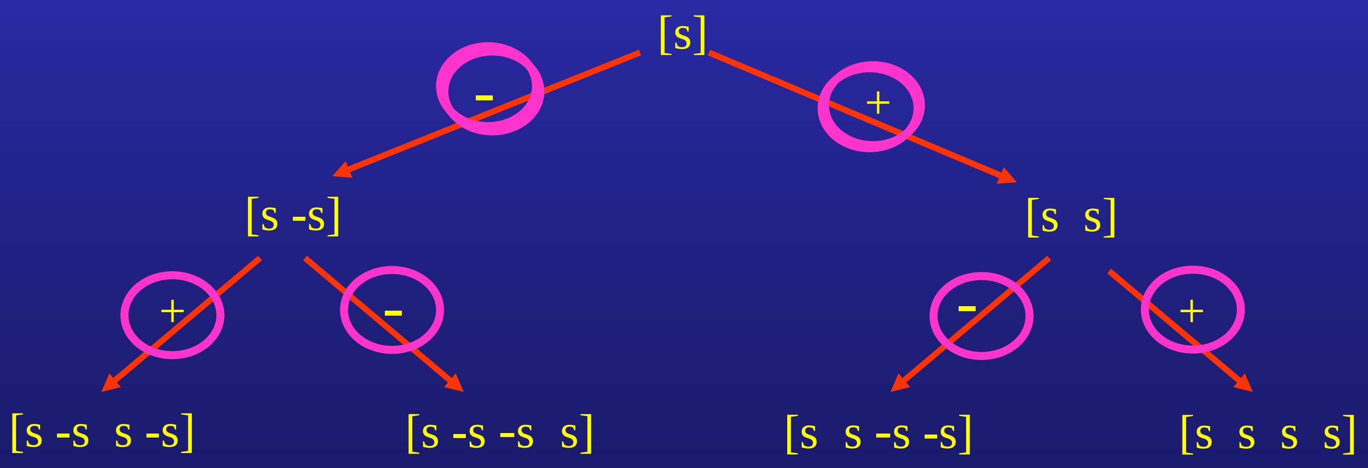


- The set of kernels at level k is denoted $V_{[s]}^{(k)}$.
- The initial seed **s** can be any vector.
- $V_{[s]}^{(k)}$ forms an orthogonal set of 2^k kernels .
- When [s]=1, $V_{[s]}^{(k)}$ forms the WH kernels of size 2^k.

Definition 1: The sequence [$\alpha_1\ \alpha_2 \ldots\ \alpha_k$] that uniquely defines a vector $v \in V_s^{(k)}$ is called the **alpha-index** of *v*.

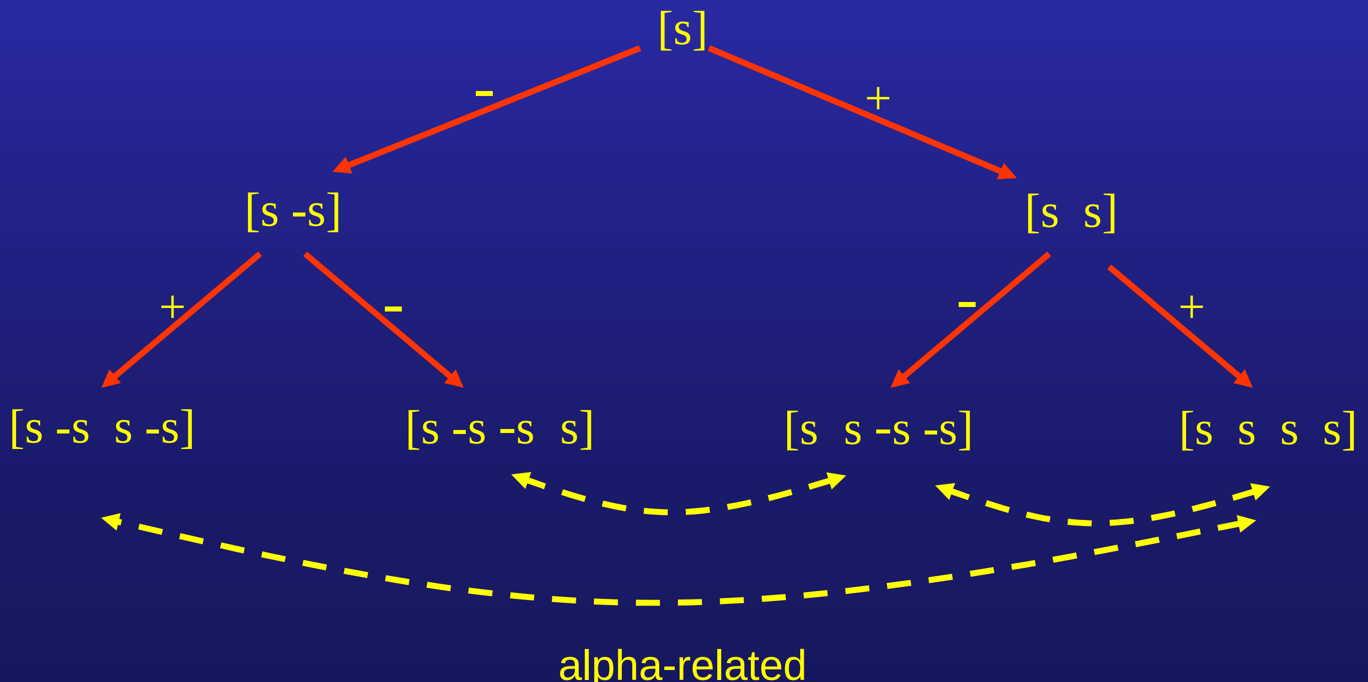
α-index: [-,+]

α-index: [+,+]



Definition 2: Two vectors $v_i, v_j \in V_s^{(k)}$ are called **alpha-related** if the hamming distance of their alpha-index is one.

An ordered set of GCK that are consecutively alpha-related are called a **Gray-Code Sequence (GCS)**



GCS Properties

Let V_+ and V_- be two α-related vectors:

V_+ and V_- share a similar prefix vector of length Δ.

$[\ldots+\ldots\ldots] \Rightarrow V_+$ $\qquad$ $[\ldots-\ldots\ldots] \Rightarrow V_-$

$\alpha_1=(+,-,-) \Rightarrow V_-$

$\alpha_2=(+,-,+) \Rightarrow V_+$

$$\alpha\text{-related} \left\{ \begin{array}{l} V_+ - [\,\boxed{s\ \ s\ -s\ -s}\ \ s\ \ s\ -s\ -s\,] \\ V_- - [\,\boxed{s\ \ s\ -s\ -s}\ -s\ -s\ \ s\ \ s\,] \end{array} \right.$$

Shared prefix, $\Delta=4|s|$

GCS Properties

Define:

$$V_p = V_+ + V_-$$

$$V_m = V_+ - V_-$$

Main Result:

$$V_p(i-\Delta) = V_m(i)$$

(Proof by induction)

Example

V_+ - [s s -s -s s s -s -s]

V_- - [s s -s -s -s -s s s]

V_p - [2s 2s -2s -2s 0 0 0 0]

V_m- [0 0 0 0 2s 2s -2s -2s]

GCS – Main Result

$$V_p(i-\Delta) = V_m(i)$$

$$V_+(i) = V_+(i-\Delta)+V_-(i)+V_-(i-\Delta)$$

$$V_-(i) = -V_-(i-\Delta)+V_+(i)-V_+(i-\Delta)$$

Efficient convolution using GCS

- If V_+ and V_- are α-related and $S(i)$ is a given signal:

$$b_+ = V_+ * S$$
$$b_- = V_- * S$$

$$b_+(i) = b_+(i-\Delta) + b_-(i) + b_-(i-\Delta)$$
$$b_-(i) = -b_-(i-\Delta) + b_+(i) - b_+(i-\Delta)$$

Given the convolution result of b_-, the convolution result of b_+ can be computed using only 2 ops/pix regardless the size of the kernels !

Example

$$\overbrace{[+1\ +1\ -1\ -1]}^{V_+}\quad \overbrace{[+1\ -1\ -1\ +1]}^{V_-}\qquad \Delta=1$$

$$b_+(i) = b_+(i-1) + b_-(i) + b_-(i-1)$$

Signal S	2	1	7	8	3	7	9	11	23	31
b_+ by GCK	-1	-1	-1	-1	-1	+1	+1			
b_-	2	-11	3	7	-2	10	6			
b_+	12	3	-5	5	10	18	-34			

2 ops/pixel regardless of size & dimension of GCK

Generalization to higher dim.

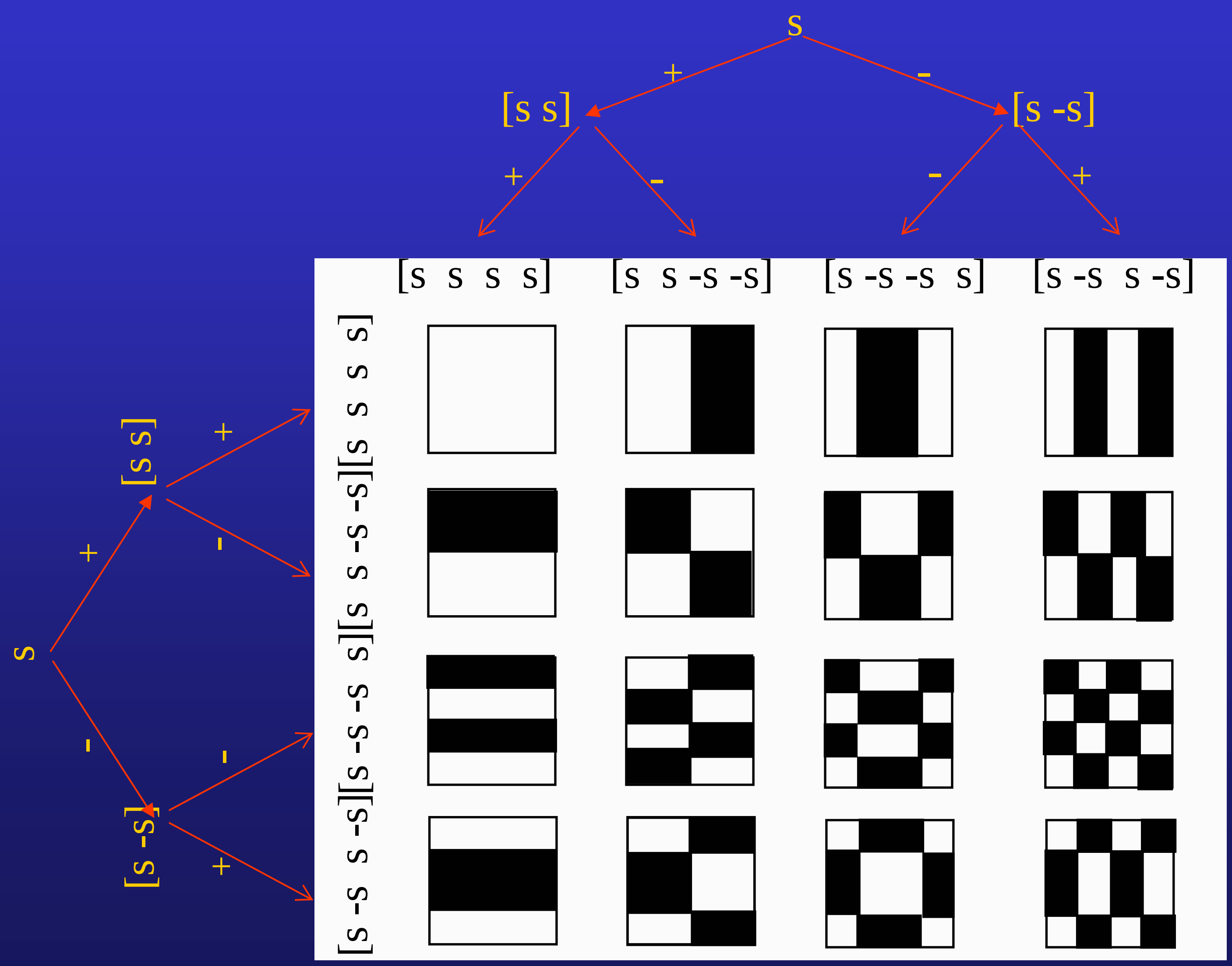
- A set of 2D kernels can be generated using an outer product of two 1D GCK

$$V_{s_1,s_2}^{(k_1,k_2)} = \left\{ v_1 \times v_2 \,\middle|\, v_1 \in V_{s_1}^{k_1}, v_2 \in V_{s_2}^{k_2} \right\}$$

$$v = v_1 \times v_2 \quad \Leftrightarrow \quad v(i,j) = v_1(i)\, v_2(j)$$

- This can be generalized to higher dimension.

Example of the set $V_{[1][1]}^{(2,2)}$ (2D WH)



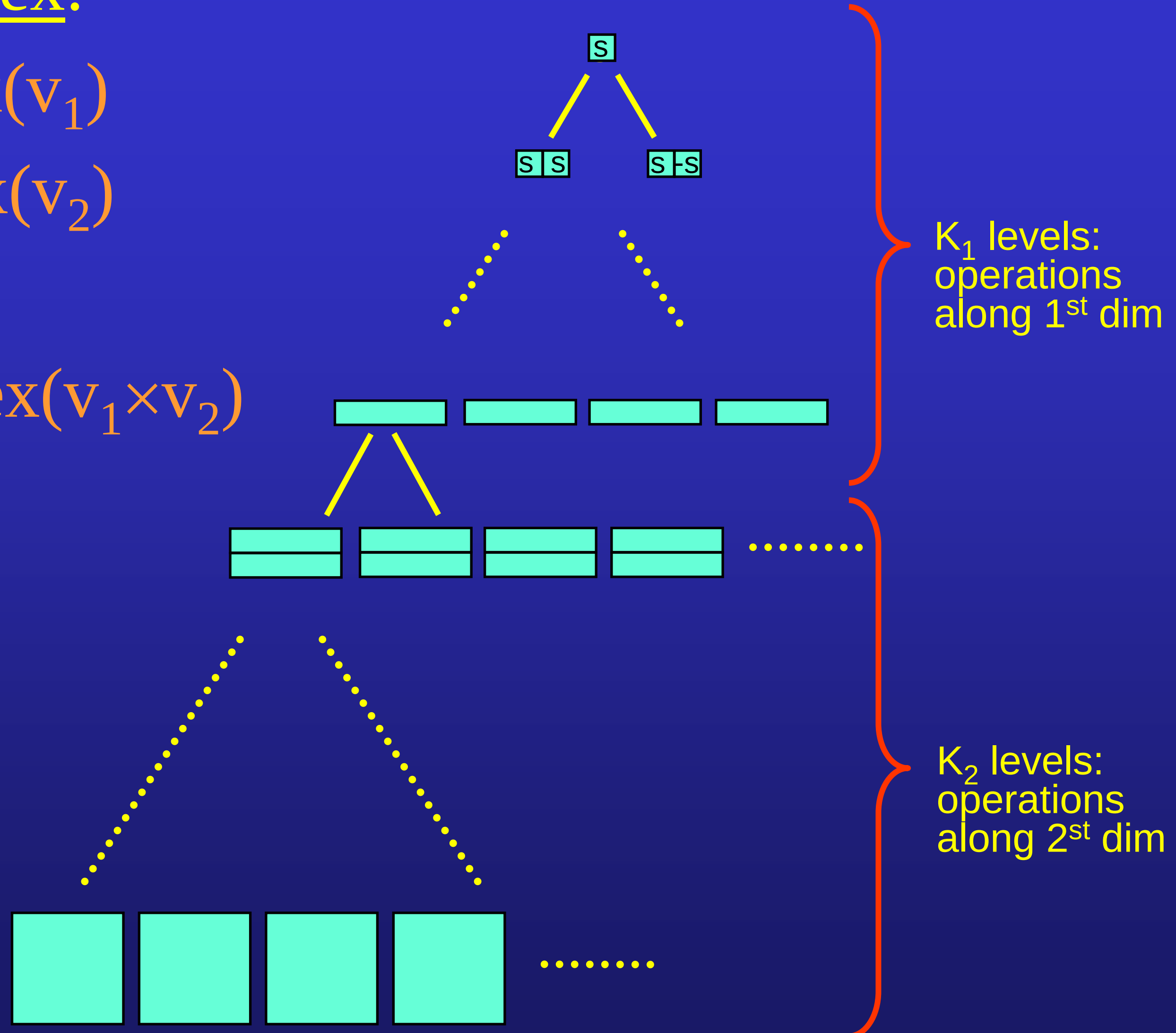
Definition α-index:

if $\underline{\alpha}_1 = \alpha\text{-index}(v_1)$

and $\underline{\alpha}_2 = \alpha\text{-index}(v_2)$

then

$$[\underline{\alpha}_1\ \underline{\alpha}_2\ ] = \alpha\text{-index}(v_1 \times v_2)$$



nD GCK

Definition: Two vectors $v_i, v_j \in V_{s_1,s_2}^{(k_1,k_2)}$ are called **alpha-related** if the hamming distance of their alpha-index is one.

An ordered set of 2D GCK that are consecutively alpha-related form a **Gray-Code Sequence (GCS)**

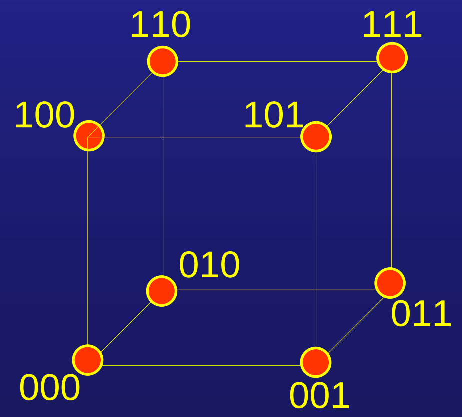
Every two consecutive 2D kernels that are α-related can be computed using only **2 ops/pix** regardless of the size (and dim.) of the kernels !

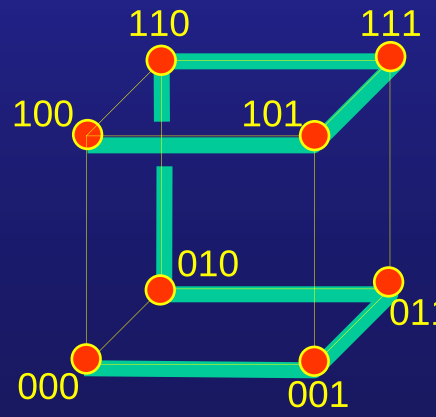
Ordering the GCS

Conclusion: Applying successive convolutions with a set of GCS kernels requires 2 ops/pixel/kernel.

- **Questions**:
  - How many GCS are there?
  - How should we choose the best GCS?

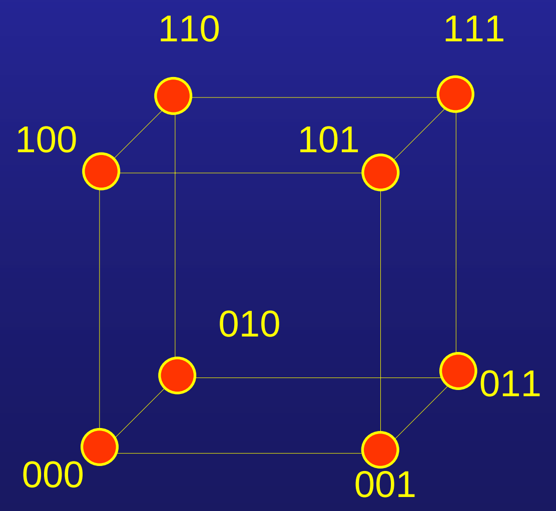
- Observation 1: The α-index of a 2D kernel $v \in V_{s_1,s_2}^{(k_1,k_2)}$ can be viewed as a vertex point in a k_1+k_2 dim hypercube.
- Observation 2: The set $V_{s_1,s_2}^{(k_1,k_2)}$ is isomorphic to a k_1+k_2 dim hypercube graph whose edges connect α-related vertices.
- Observation 3: A GCS is isomorphic to a Hamiltonian path in the hypercube graph.





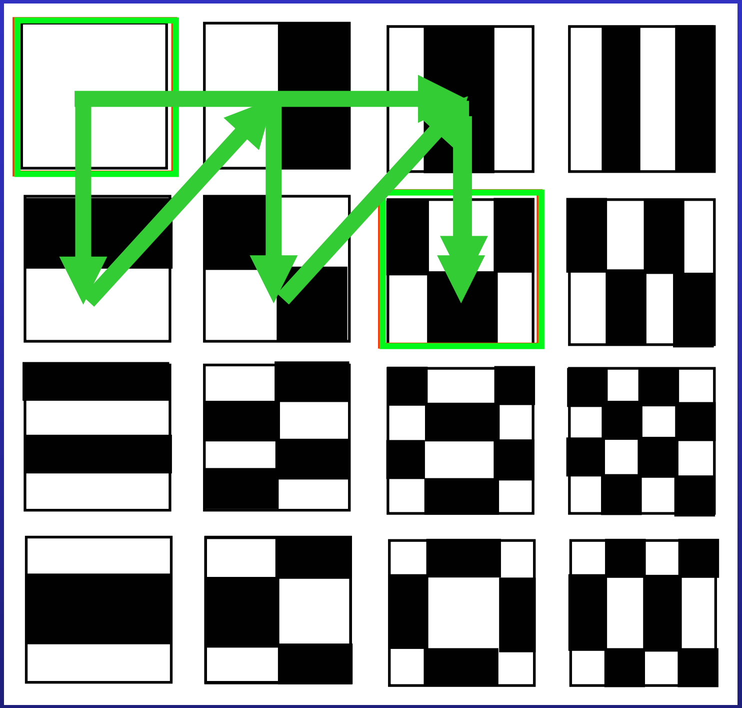
- Conclusion 1: The number of possible GCS is identical to the number of different Hamiltonian cycles in the associated hypercube graph (2, 8, 96, 43008, ... [Gardner 86]) .

- Conclusion 2: Finding an optimal GCS is NP-Complete.



Example

We would like to convolve with the marked WH kernels:

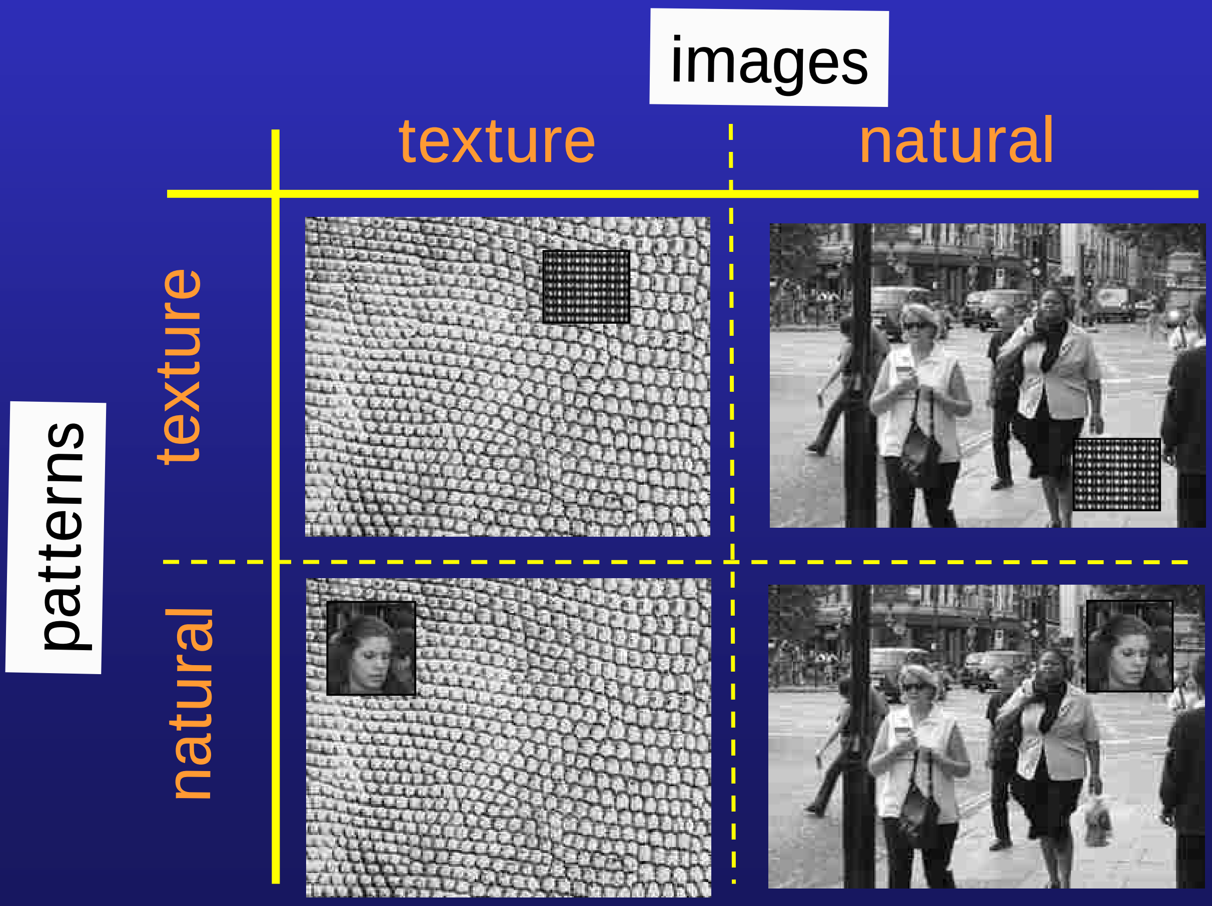


Schemes	Greedy	: O(logk)
	Sequency	: O(1)-O(logk)
	GCS	: O(1)

kernel/pixel.

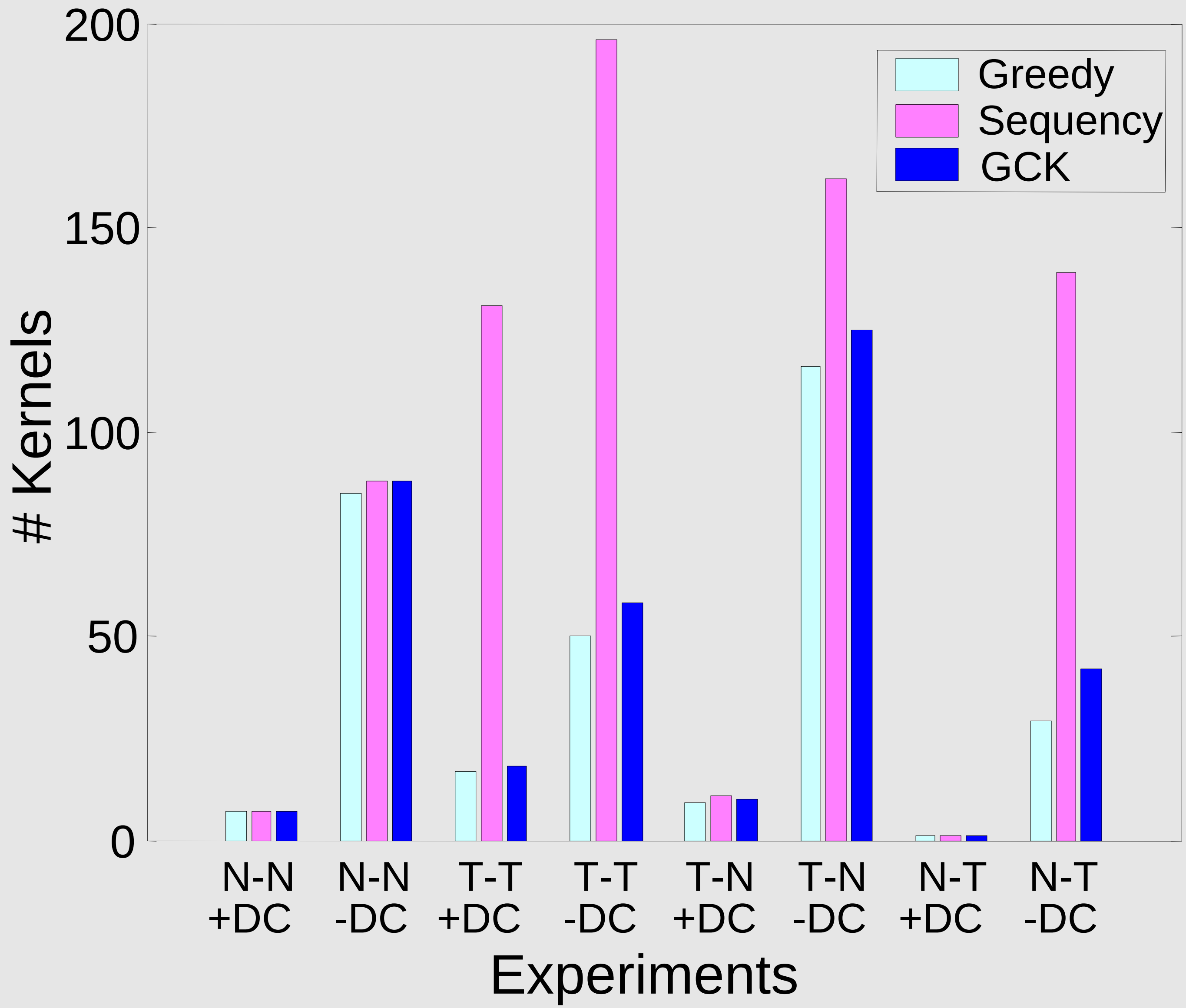
Experiments

- Task: pattern matching using WH projection kernels (*Hel-Or et. Al. 2003*).
- Measure total number of operations with and without DC.

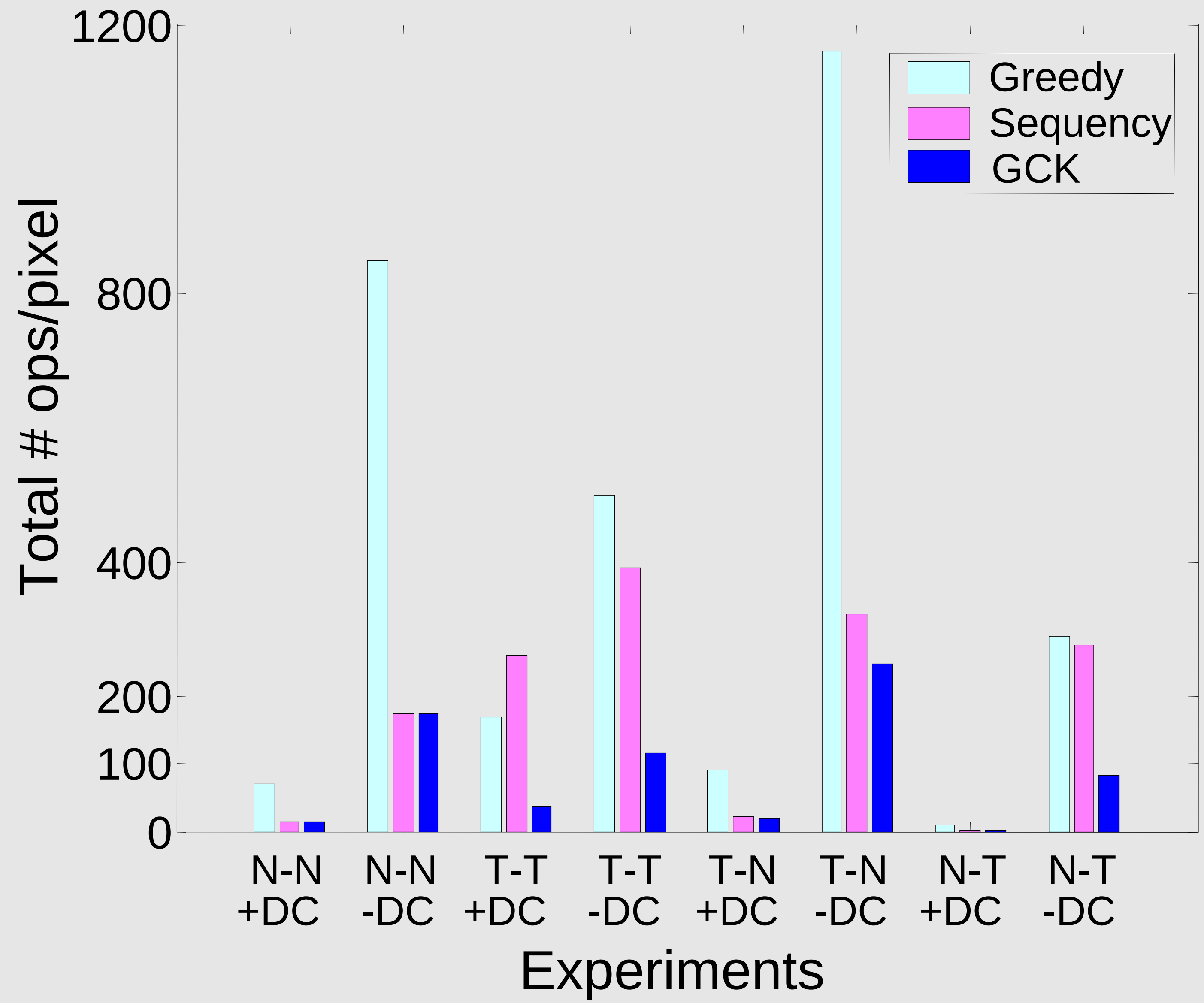


512 x 512 images

32 x 32 pattern

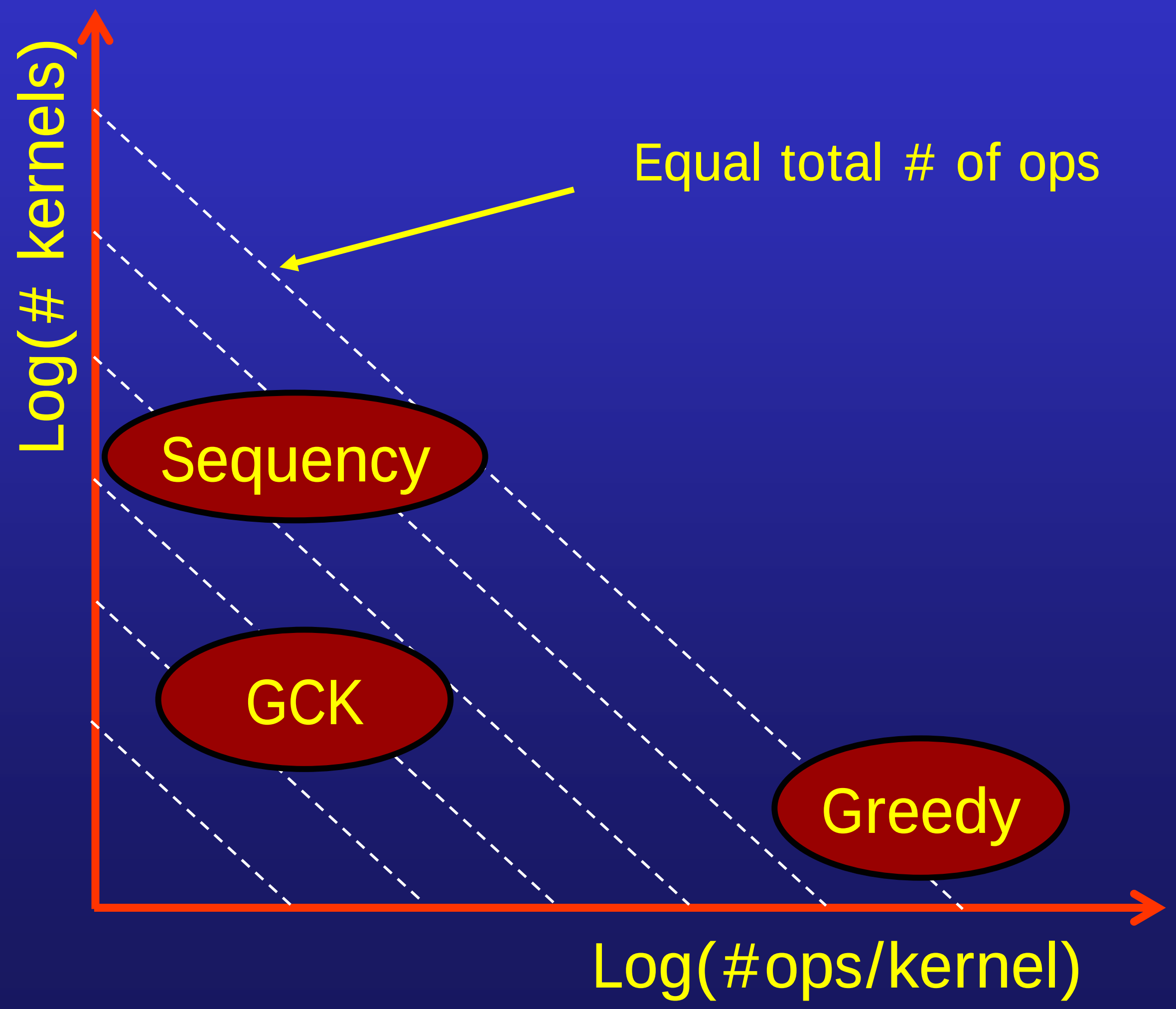


Greedy
Sequency
GCK
200
150
100
50
0
Kernels
N-N +DC
N-N -DC
T-T +DC
T-T -DC
T-N +DC
T-N -DC
N-T +DC
N-T -DC
Experiments


Total # ops/pixel
1200
800
400
200
100
0
Greedy
Sequency
GCK
N-N +DC
N-N -DC
T-T +DC
T-T -DC
T-N +DC
T-N -DC
N-T +DC
N-T -DC
Experiments

Experiments-summary

Total # of ops = (# kernels) * (# ops/kernels)



Conclusions

Advantages

- Highly efficient – 2 ops/pixel/kernel.
- Independent of the kernel size and dimension – depends only on the number of kernels.
- Integer operations.
- Very large set of kernels, using flexible design.
- The order of kernels can be optimized to include informative kernels (NP complete).
- Requires only 2|image| memory size.

Limitations

- Each kernel - computation depends on the previous kernels in the sequence. For a single kernel this framework is inefficient.
- The kernels cannot be computed using ANY order that we choose.
- Efficient only when used on a group of image windows (not on a single one).

THE END