Real Time Pattern Detection

Yacov Hel-Or

The Interdisciplinary Center

joint work with

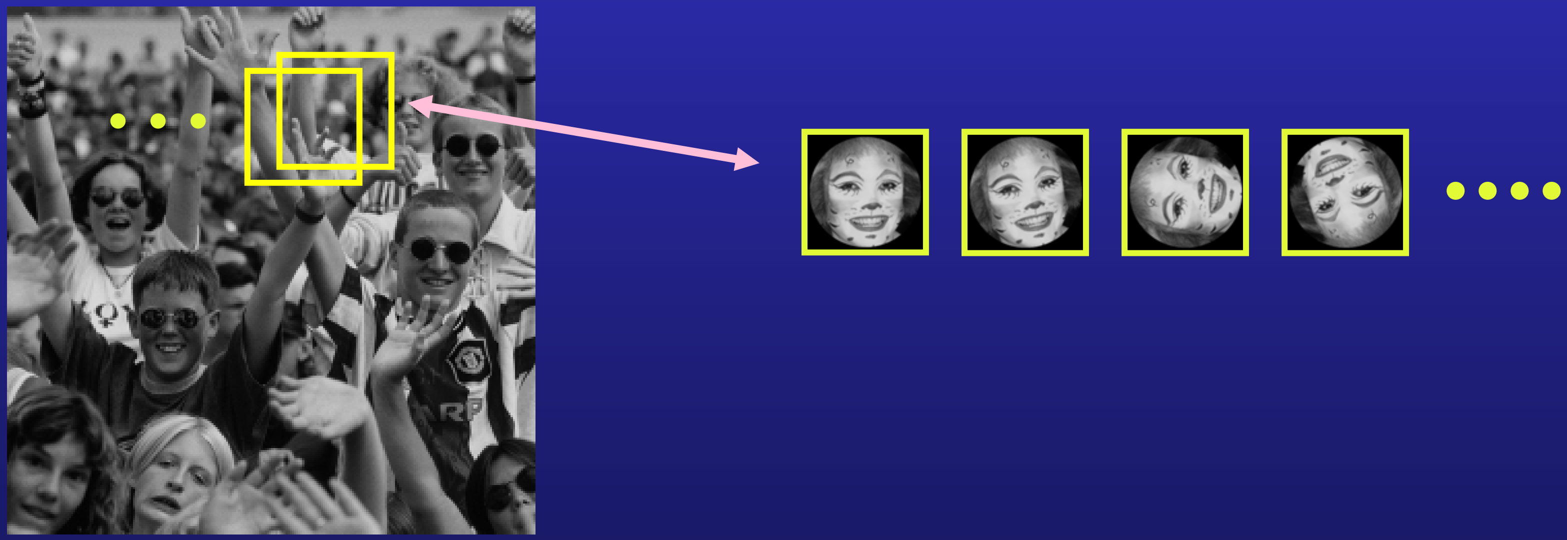
Hagit Hel-Or

Haifa University

Pattern Detection

A given pattern is sought in an image.

- The pattern may appear at any location in the image.
- The pattern may be subject to any transformation (within a given transformation group).



Example

Face detection in images

Why is it Expensive?
The search in Spatial Domain

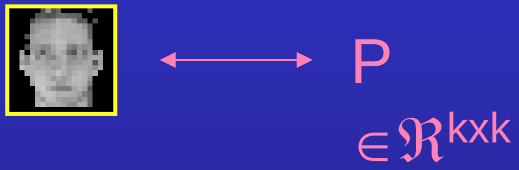
Searching for faces in a 1000x1000 image, is applied 1e6 times, for each pixel location.

A very expensive search problem

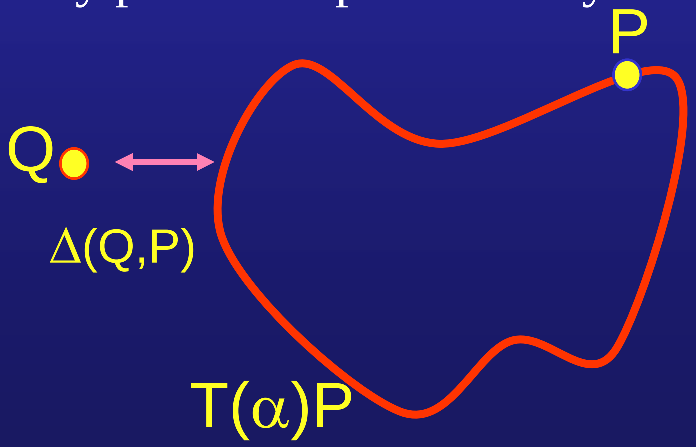
Why is it difficult?

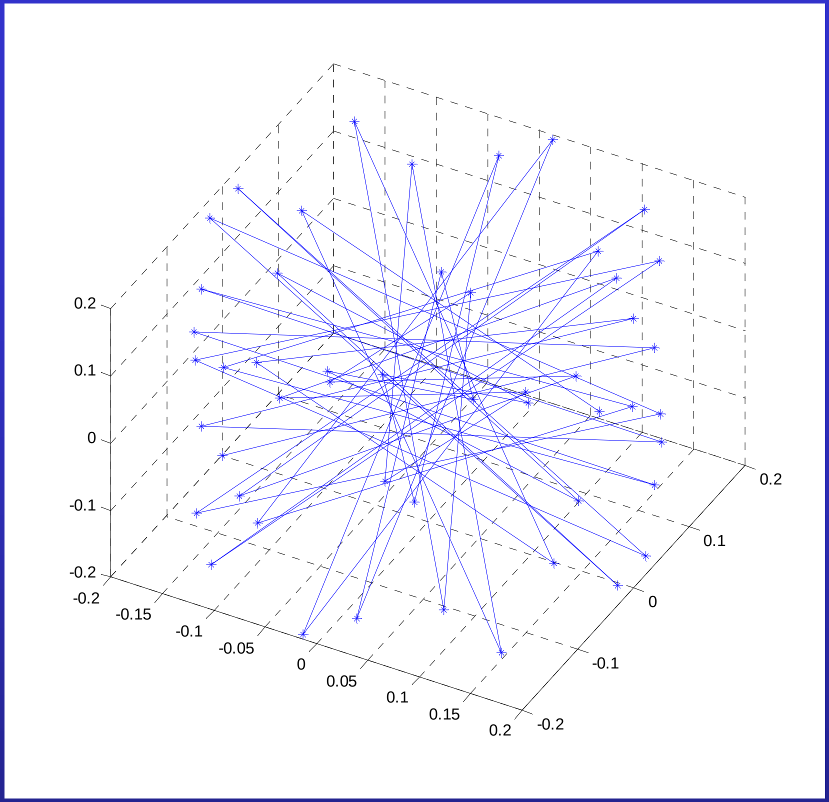
The Search in Transformation Domain

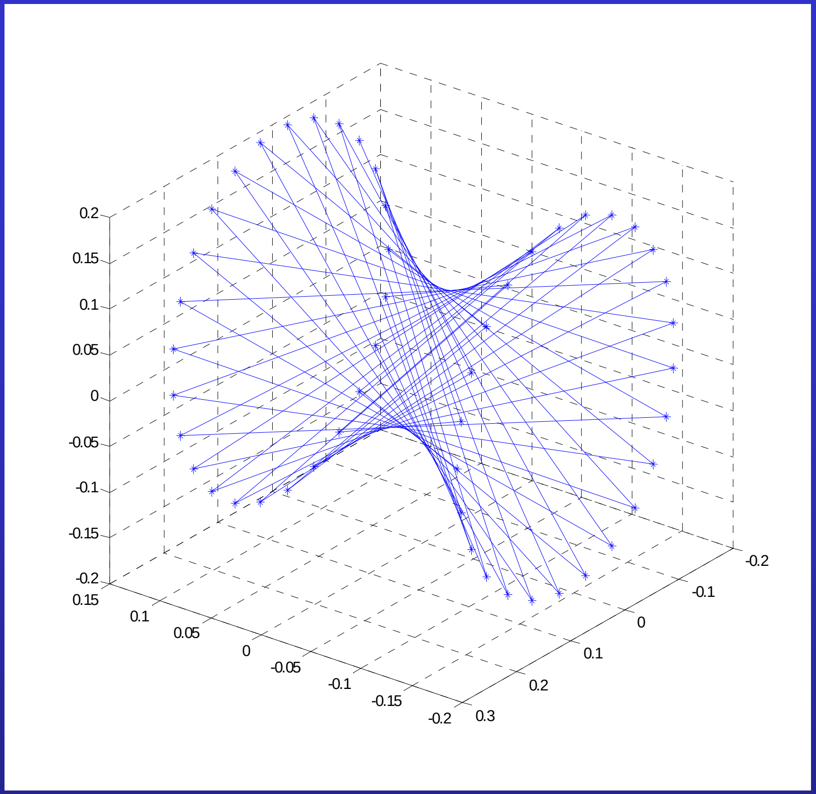
- A pattern under transformations draws a very complex manifold in “pattern space”:



  - In a very high dimensional space.
  - Non convex.
  - Non regular (two similarly perceived patterns may be distant in pattern space).







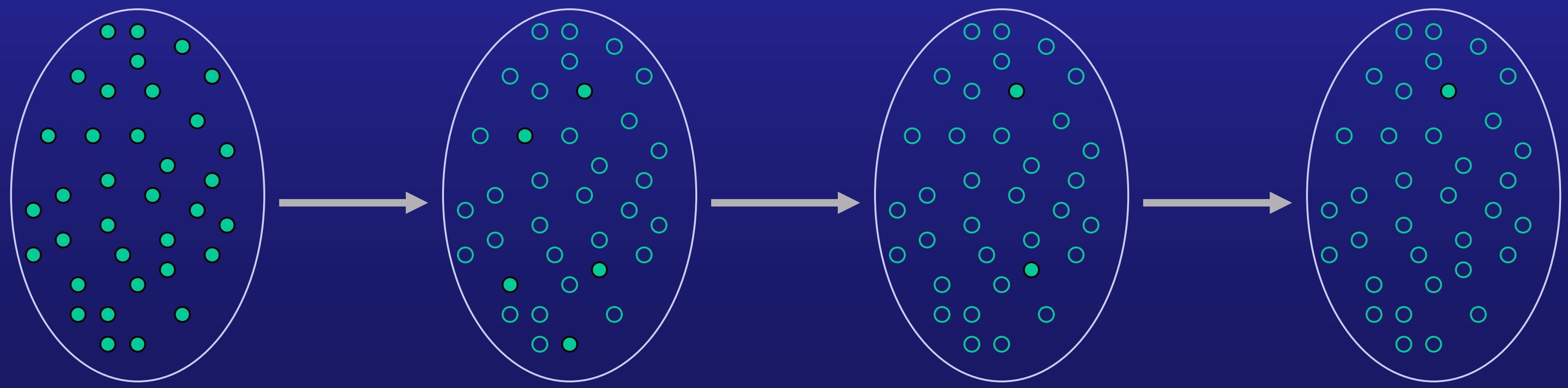
A rotation manifold of a pattern drawn in "pattern-space"
The manifold was projected into its three most significant components.

Suggested Approach

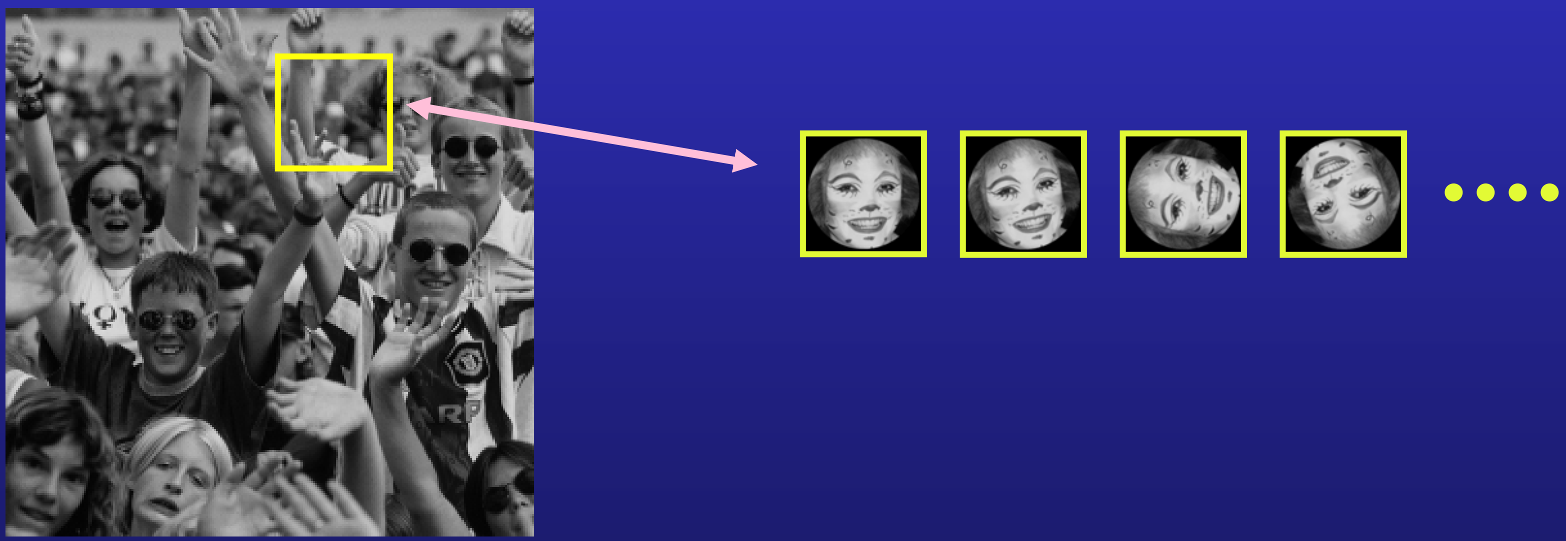
Reduce complexity of search using 2 complementary processes:

1. Reduce search in Transformation Domain.
2. Reduce search in Spatial Domain.

Both processes are based on a **Rejection Scheme**.



Efficient Search in the Transformation Domain

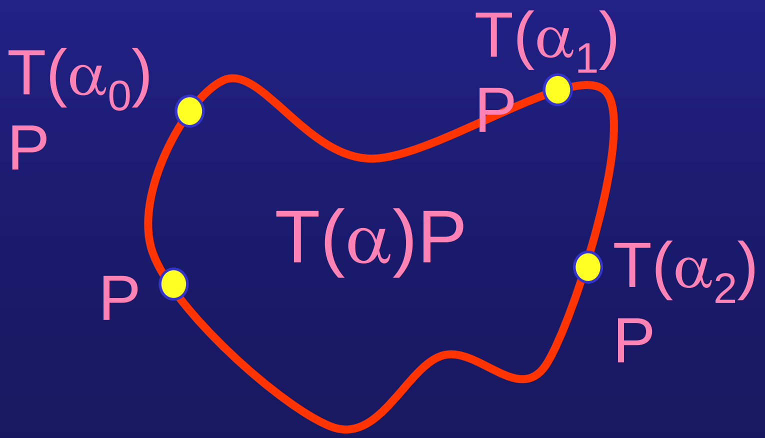


Transformation Manifold

A pattern P can be represented as a point in $\Re^{kxk}$

$T(\alpha)P$ is a transformation $T(\alpha)$ applied to pattern **P**.

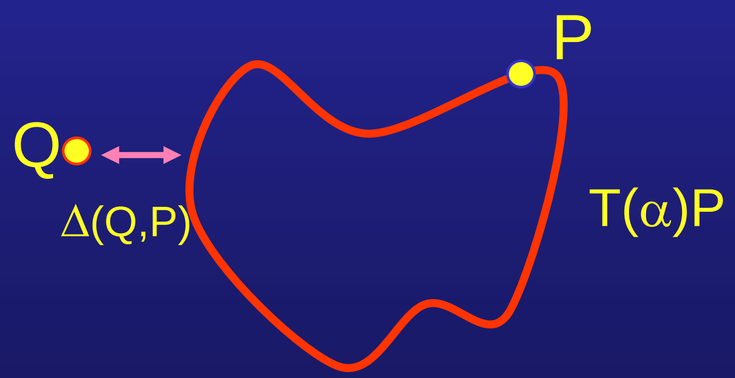
$T(\alpha)P$ for all α forms an orbit in $\Re^{kxk}$



Fast Search in Group Orbit

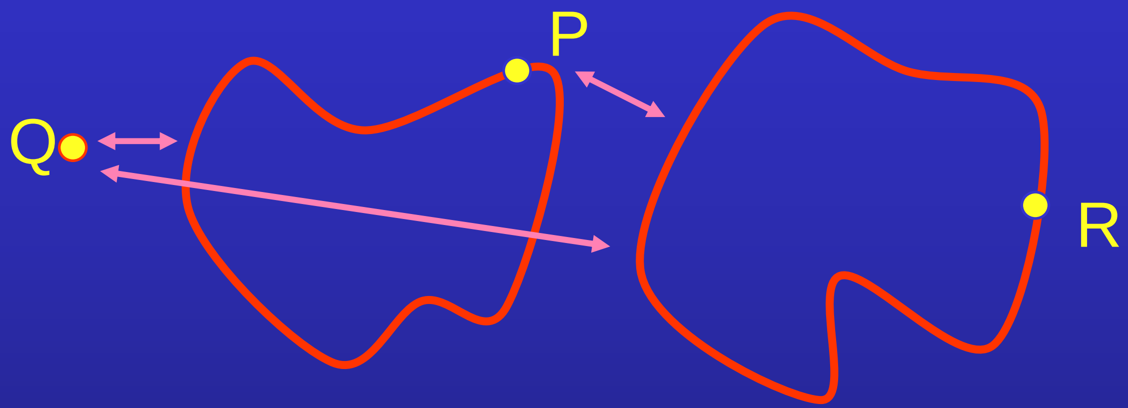
- Assume $d(Q,P)$ is a distance metric.
- We would like to find

$$\Delta(Q,P)=\min_{\alpha} d(Q, T(\alpha)P)$$



Fast Search in Group Orbit (Cont.)

- In the general case $\Delta(Q,P)$ is not a metric.

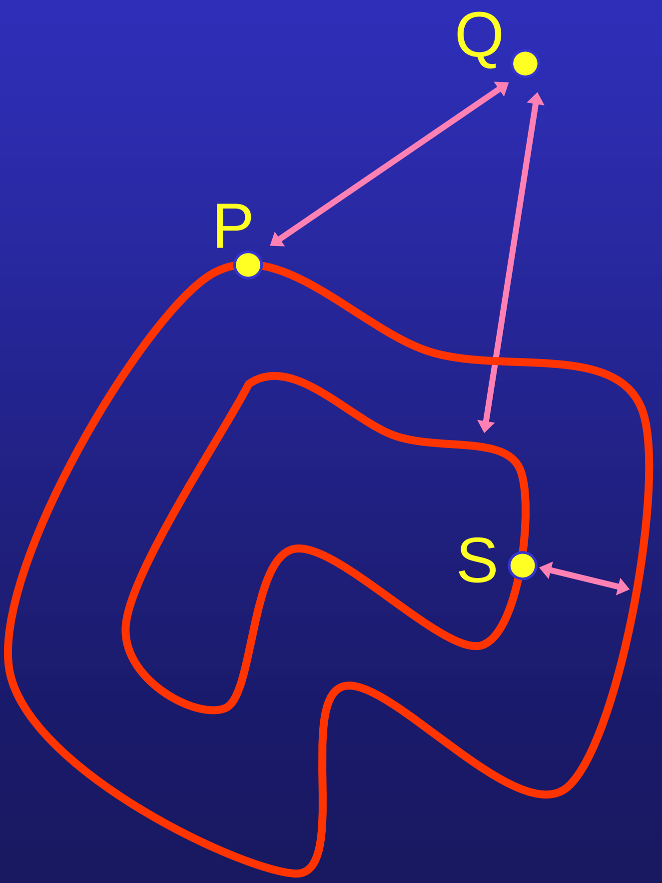


- Observation: if $d(Q,P) = d(T(\alpha)Q, T(\alpha)P)$

$\Delta(Q,P)$ is a **metric**

Fast Search in Group Orbit (Cont.)

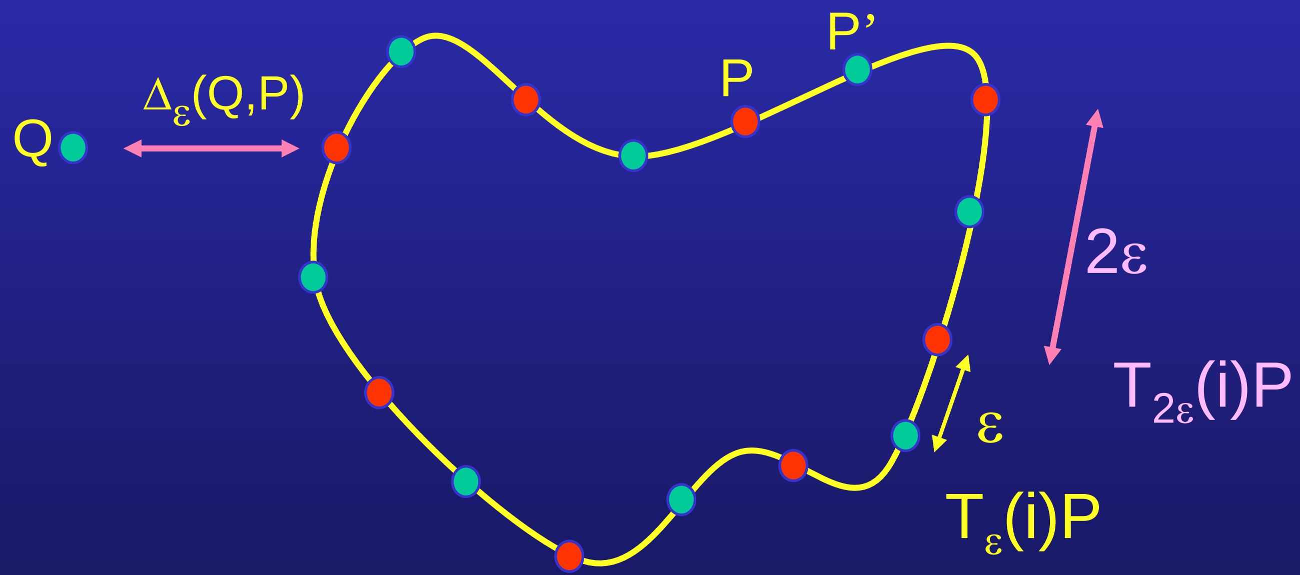
The metric property of Δ(Q,P) implies triangular inequality on the distances.



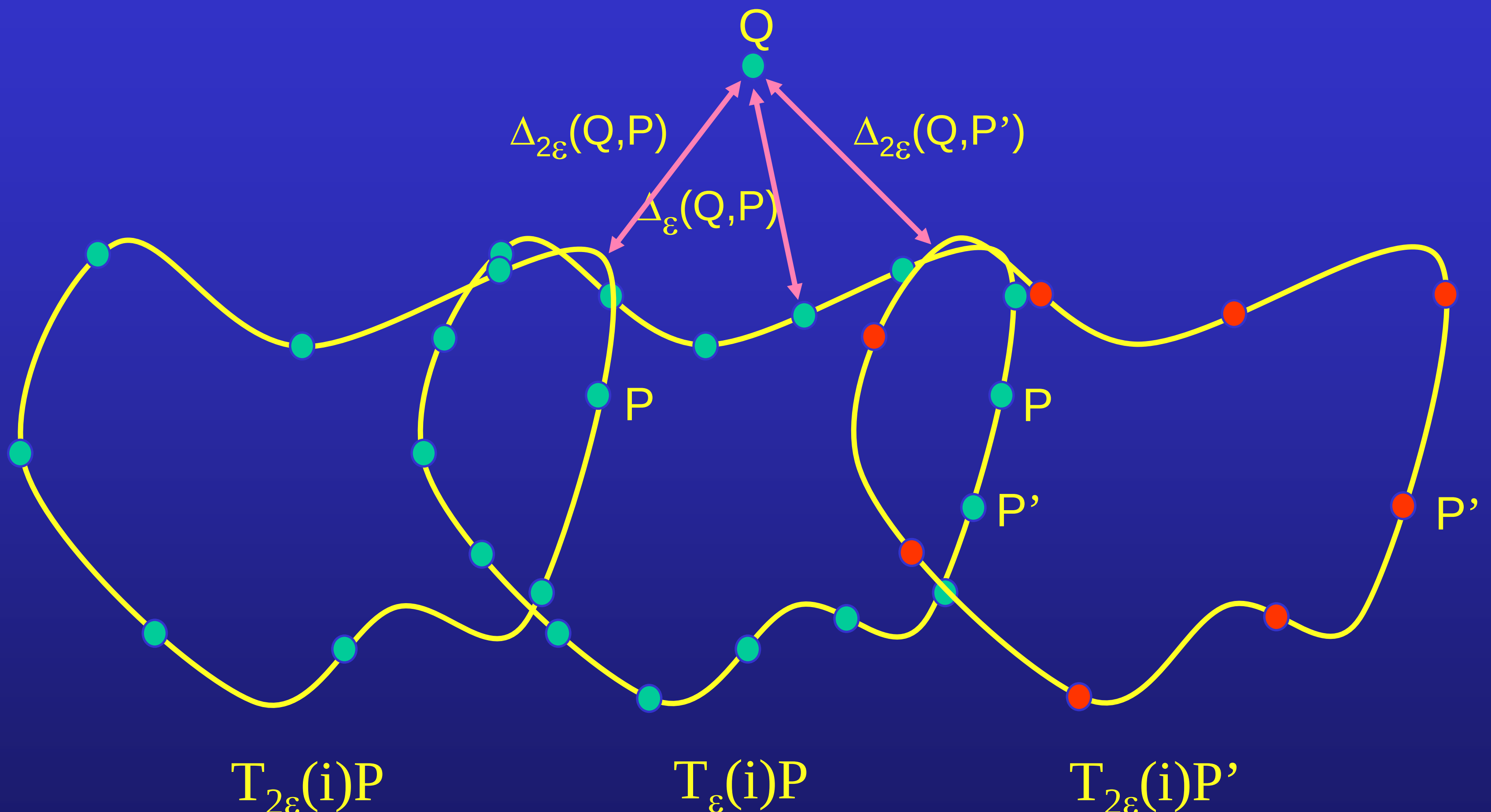
Orbit Decomposition

- In practice $T(\alpha)$ is sampled into $T(\varepsilon i)=T_\varepsilon(i)$, $i=1,2,\ldots$
- We can divide $T_\varepsilon(i)P$ into two sub-orbits:

$T_{2\varepsilon}(i)P$ and $T_{2\varepsilon}(i)P'$ where $P' = T_\varepsilon(1)\,P$

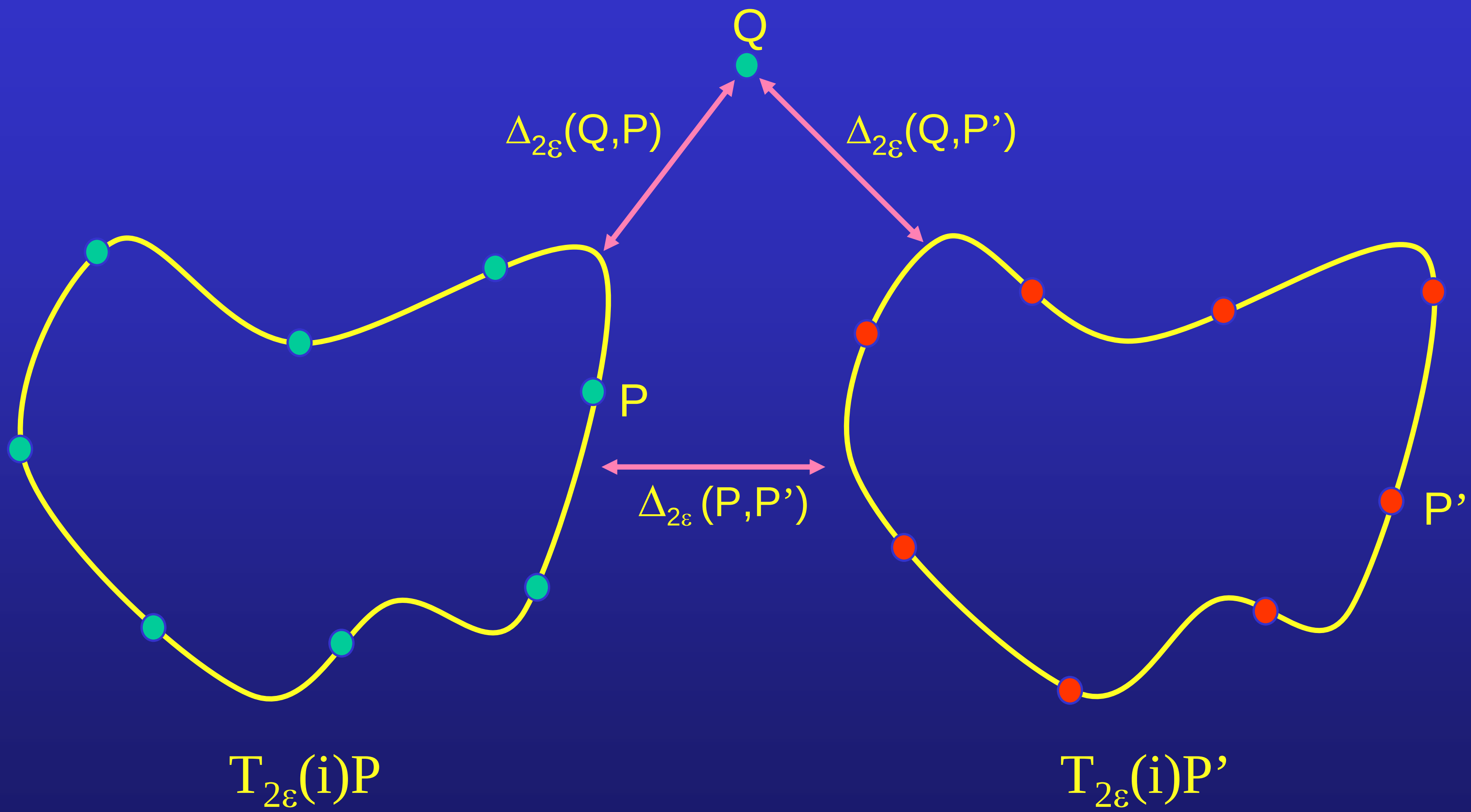


Orbit Decomposition (Cont.)



$$\Delta_{\varepsilon}(Q,P) = Min\{\Delta_{2\varepsilon}(Q,P), \Delta_{2\varepsilon}(Q,P')\}$$

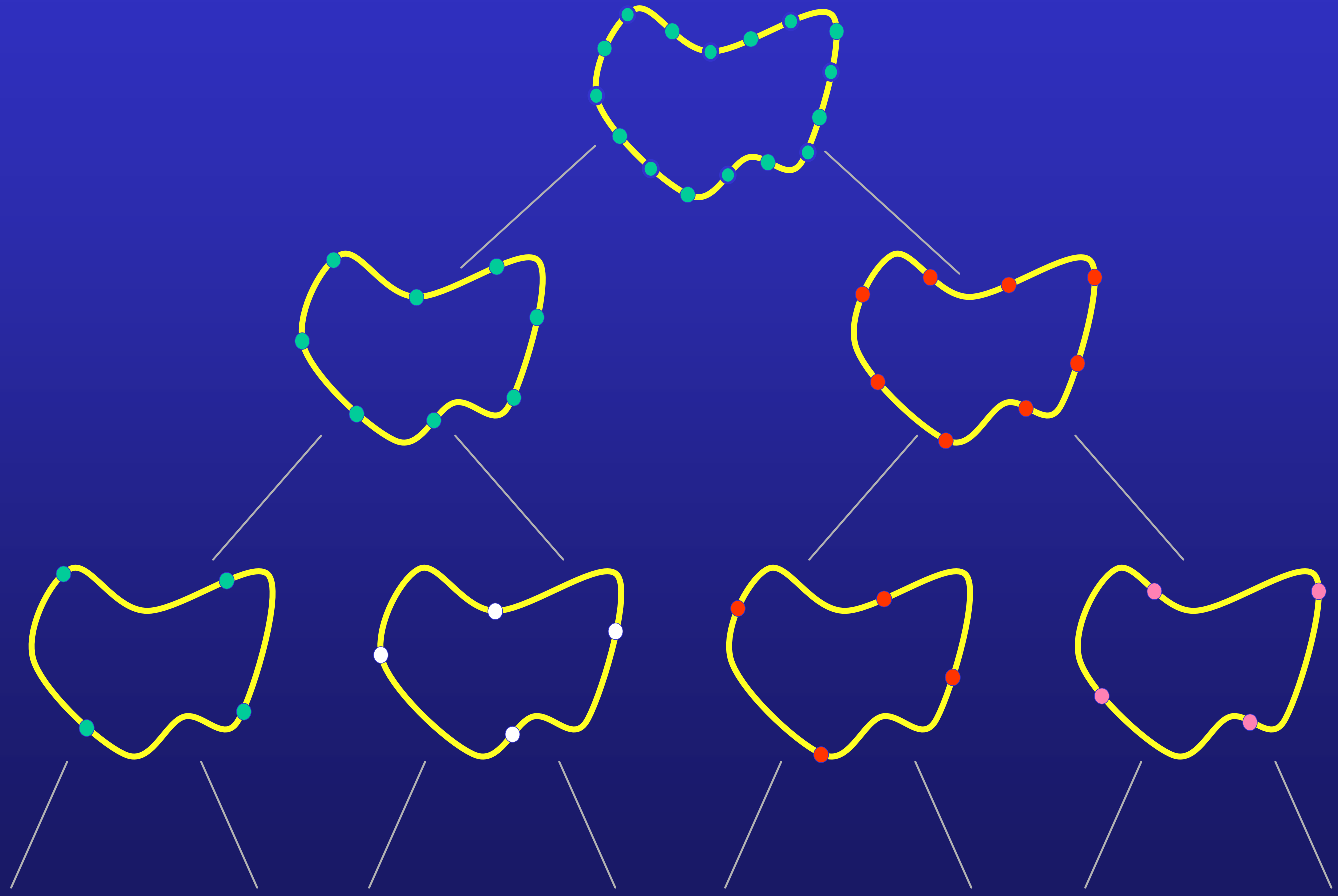
Orbit Decomposition (Cont.)



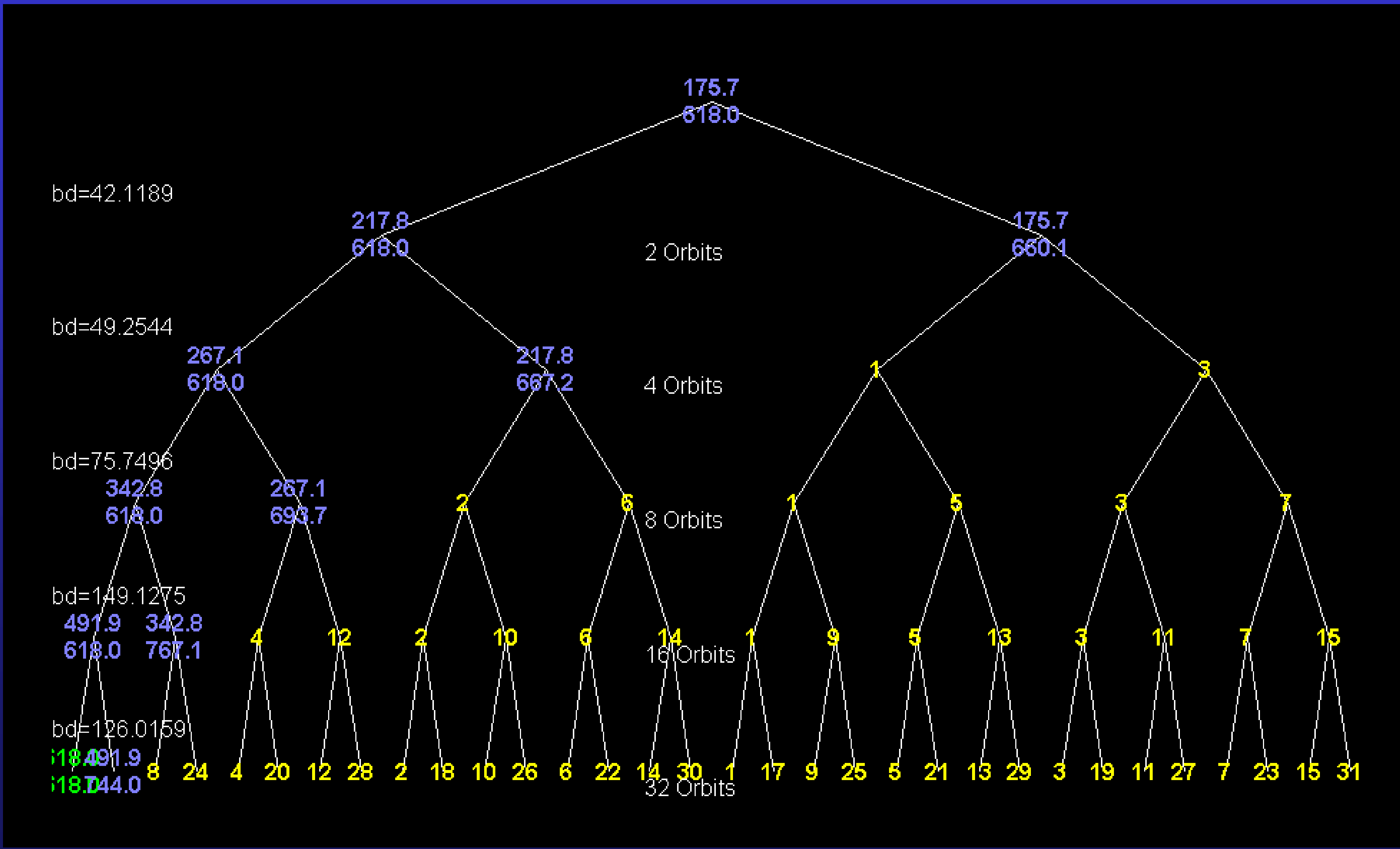
Since $\Delta_{2\varepsilon}$ is a metric and $\Delta_{2\varepsilon}(P,P')$ can be calculated in advance we may save calculations using the triangle inequality constraint.

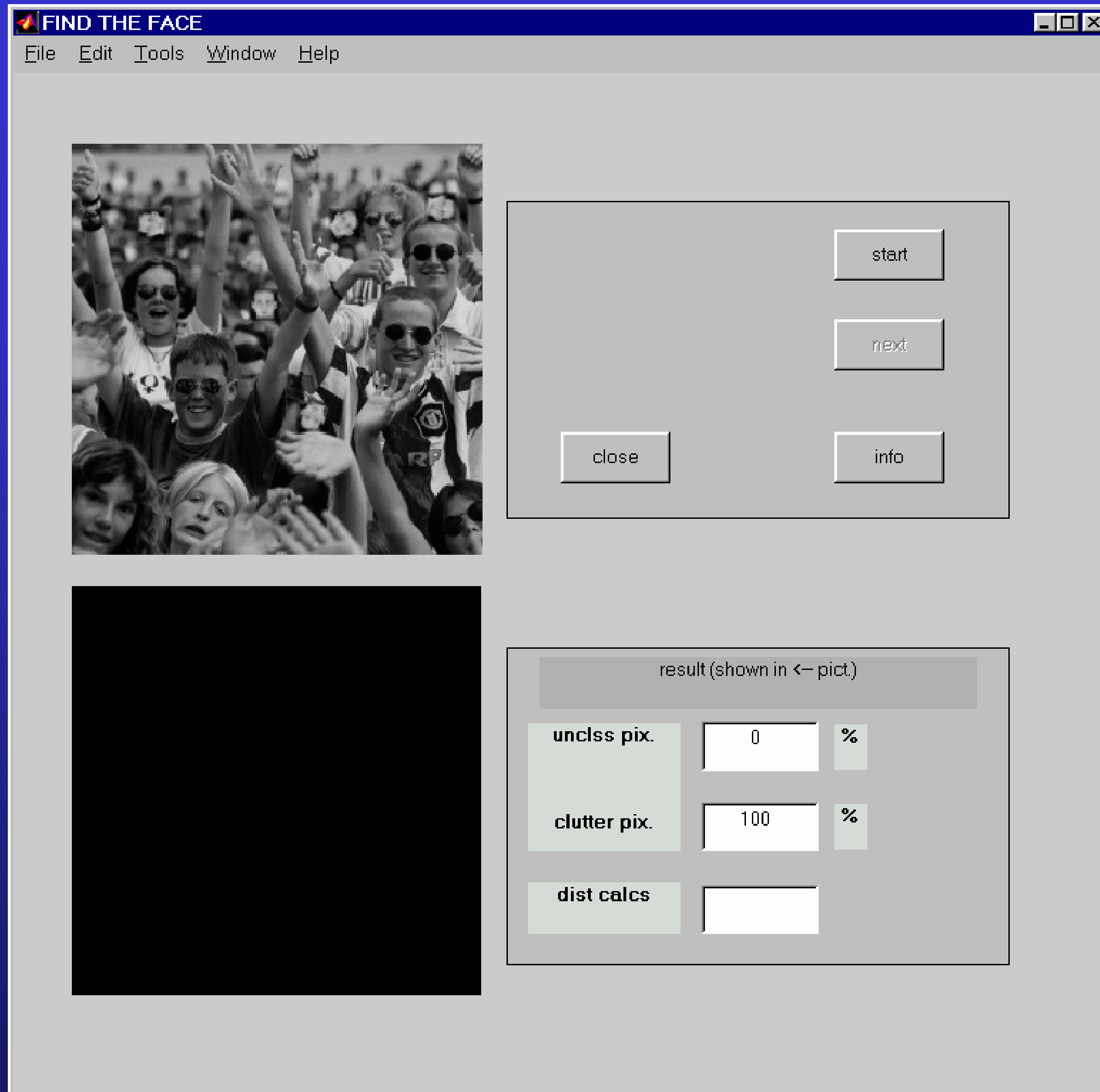
Orbit Decomposition (Cont.)

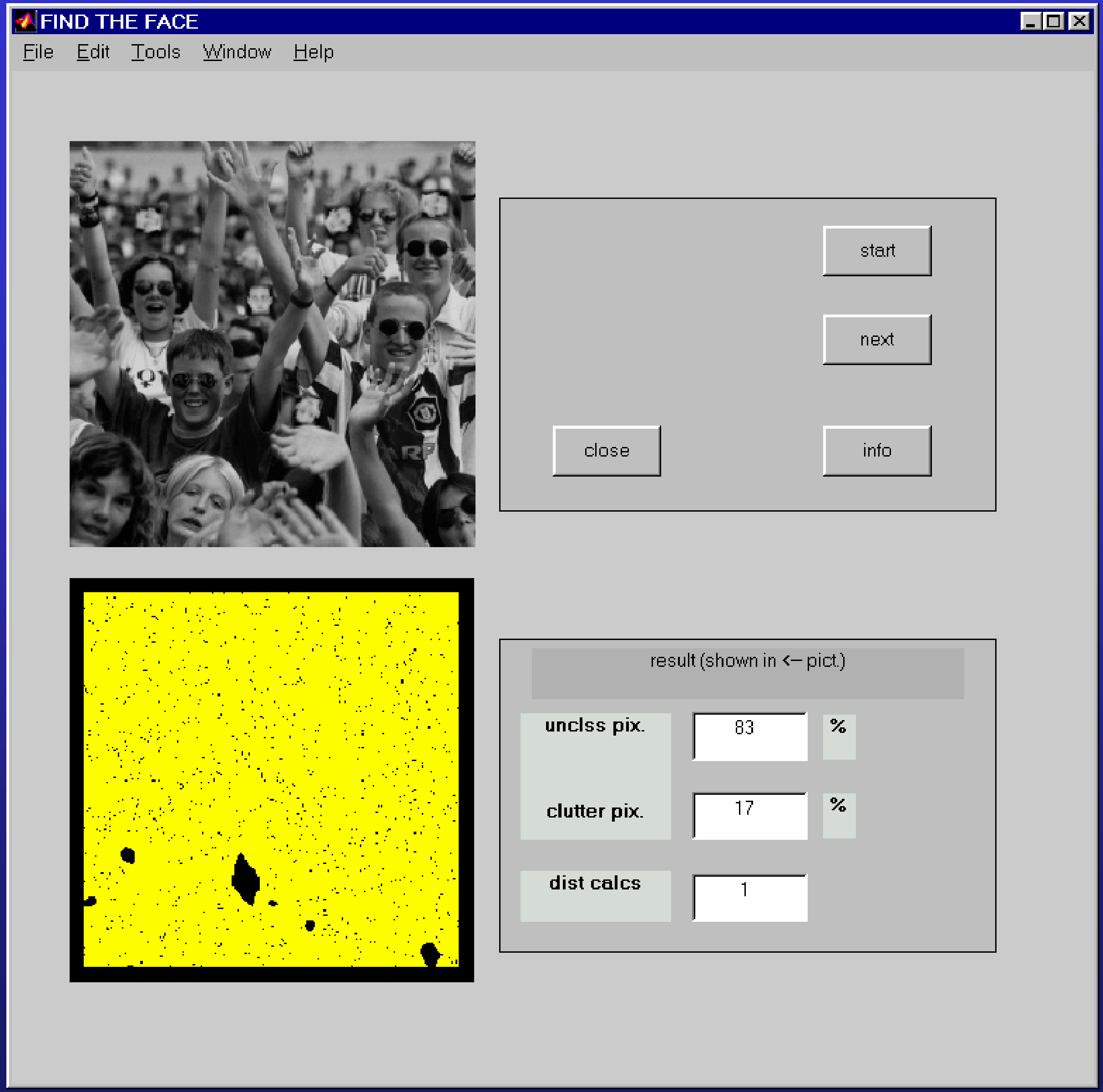
- The sub-group subdivision can be applied recursively.

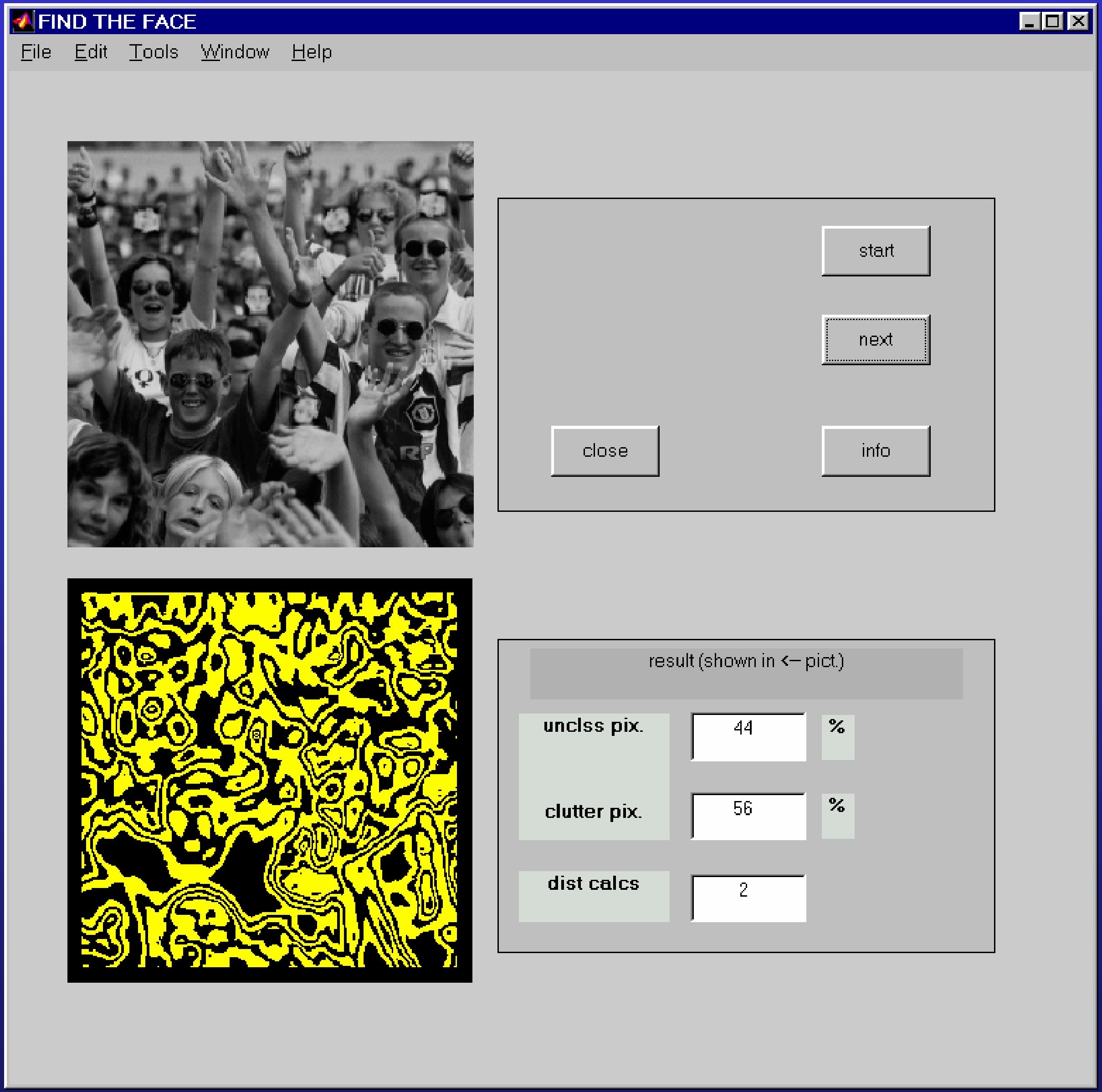


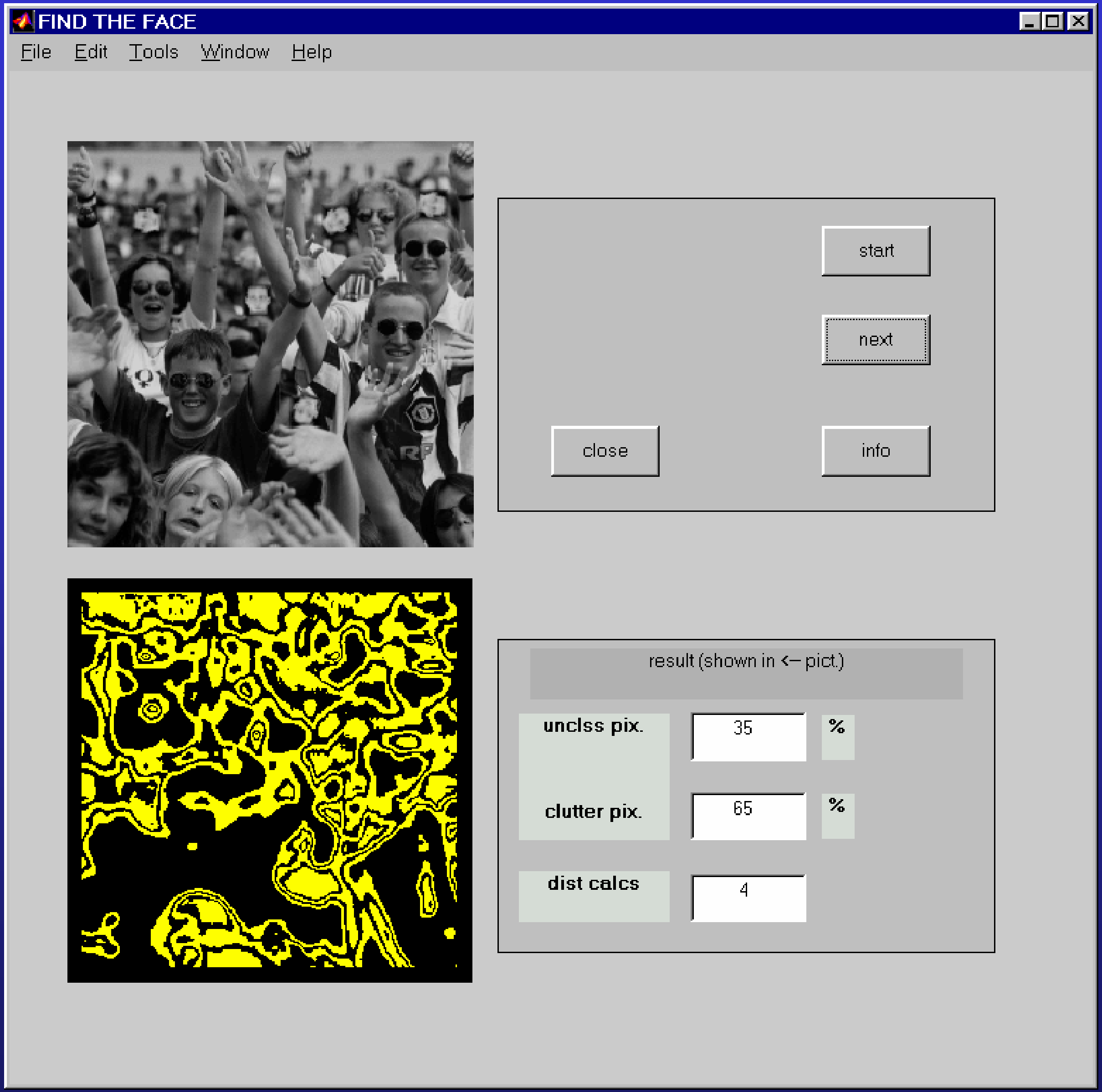
Fast Search - Example

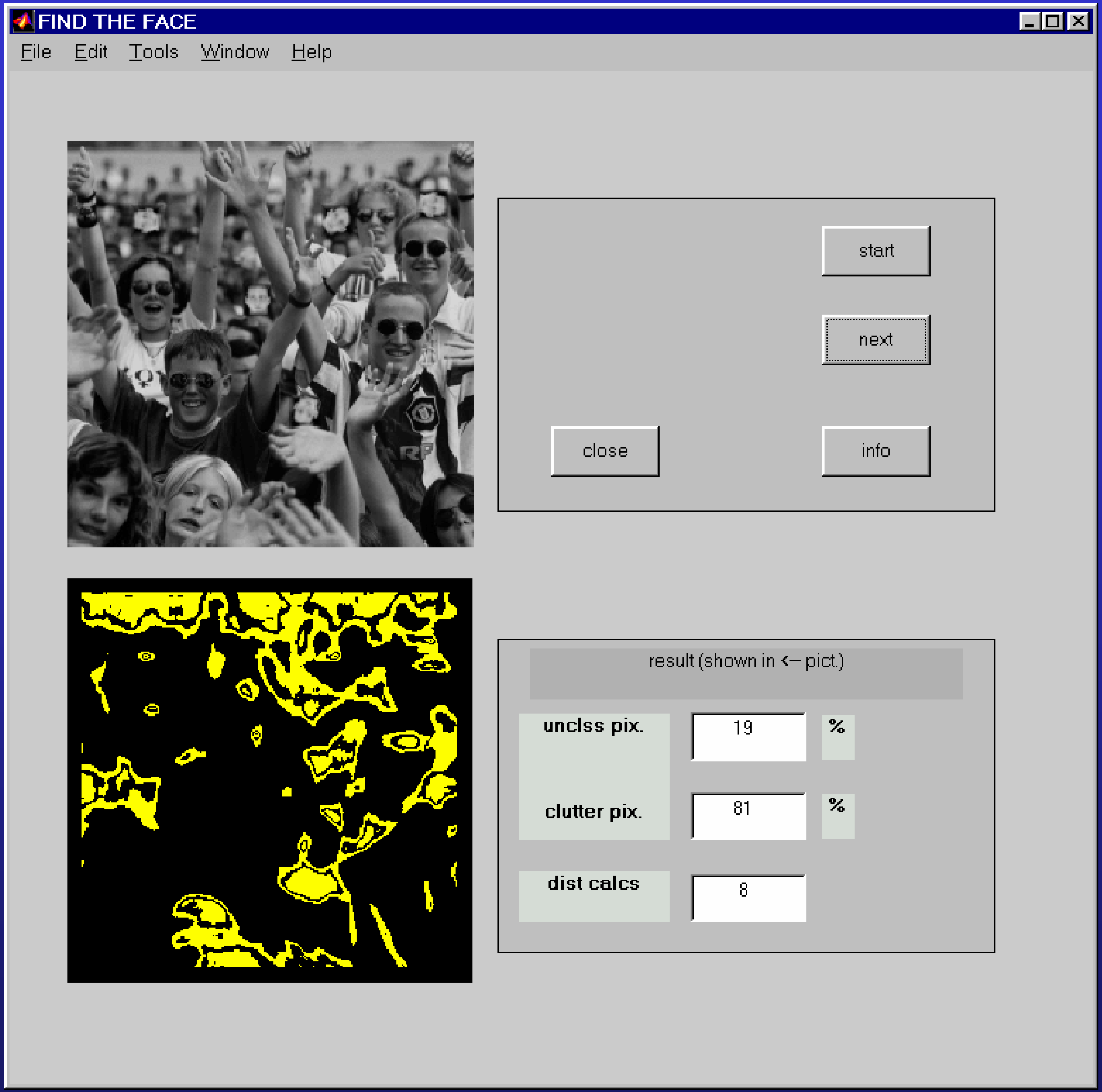


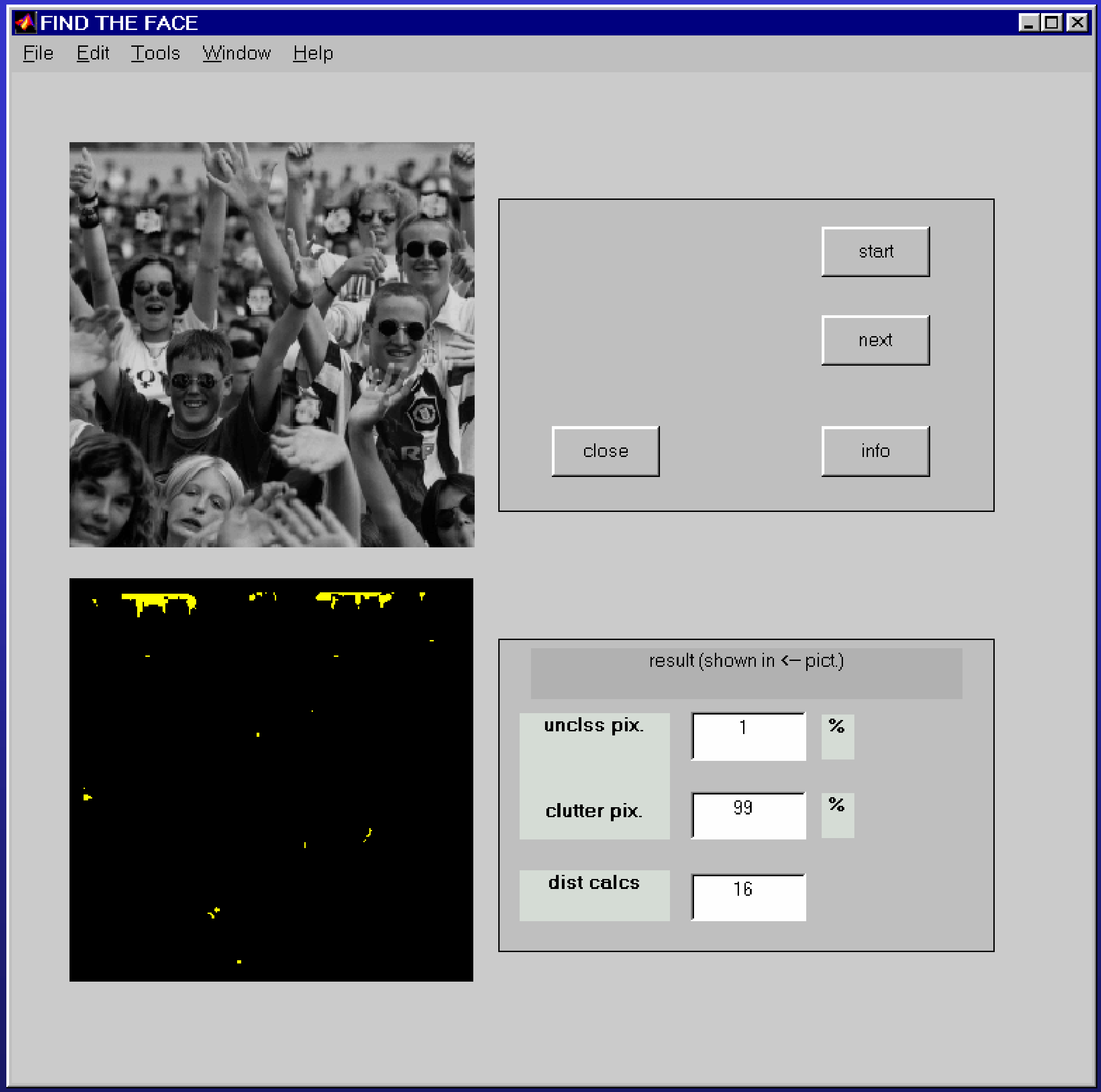

FIND THE FACE
File
Edit
Tools
Window
Help
start
next
close
info
result (shown in <– pict.)
unclss pix.
0
%
clutter pix.
100
%
dist calcs

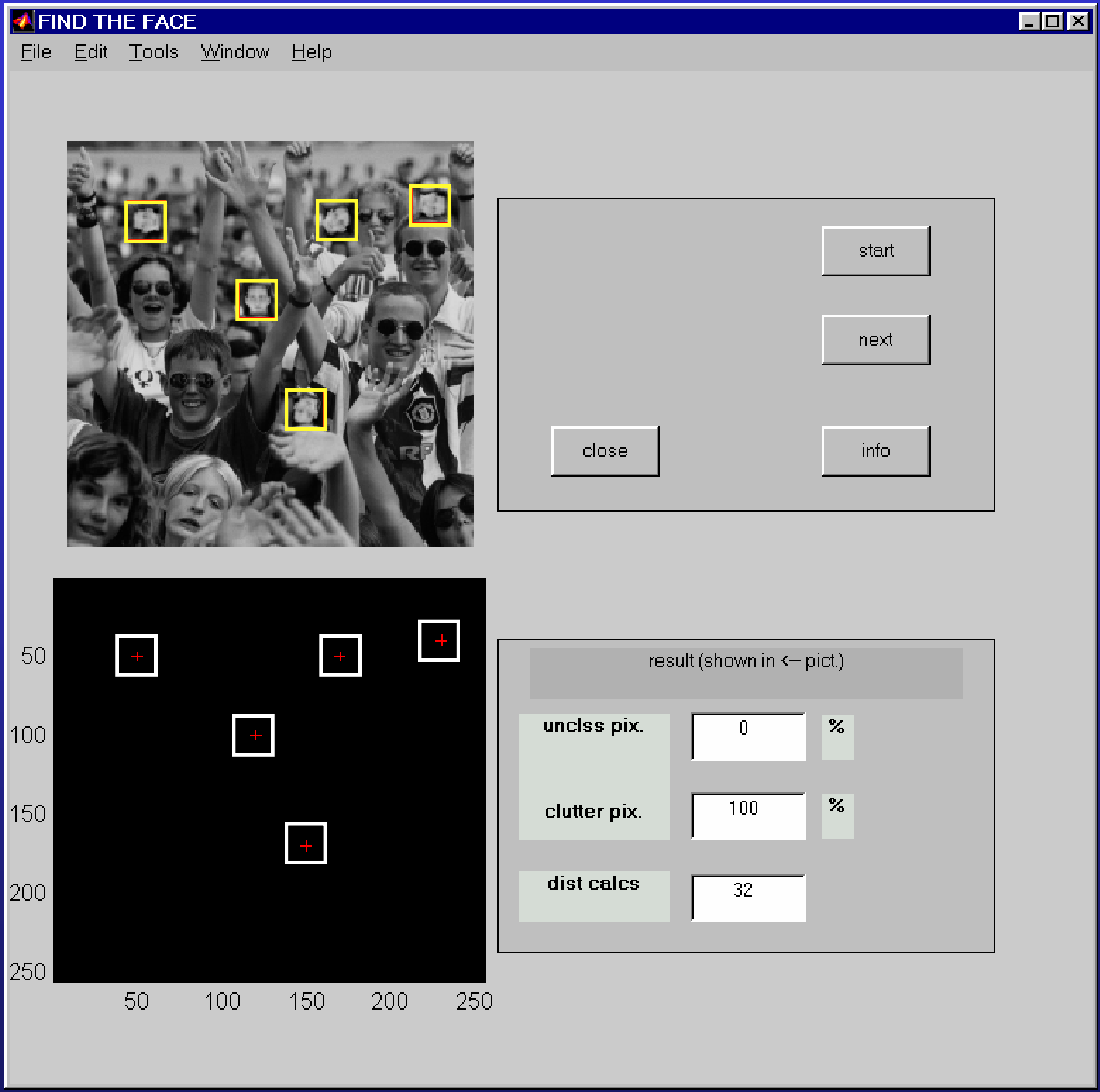

FIND THE FACE
File
Edit
Tools
Window
Help
start
next
close
info
result (shown in <– pict.)
unclss pix.
83
%
clutter pix.
17
%
dist calcs
1


FIND THE FACE
File
Edit
Tools
Window
Help
start
next
close
info
result (shown in <– pict.)
unclss pix.
44
%
clutter pix.
56
%
dist calcs
2


FIND THE FACE
File
Edit
Tools
Window
Help
start
next
close
info
result (shown in <-- pict.)
unclss pix.
35
%
clutter pix.
65
%
dist calcs
4


FIND THE FACE
File Edit Tools Window Help
start
next
close
info
result (shown in <— pict.)
unclss pix.
19
%
clutter pix.
81
%
dist calcs
8

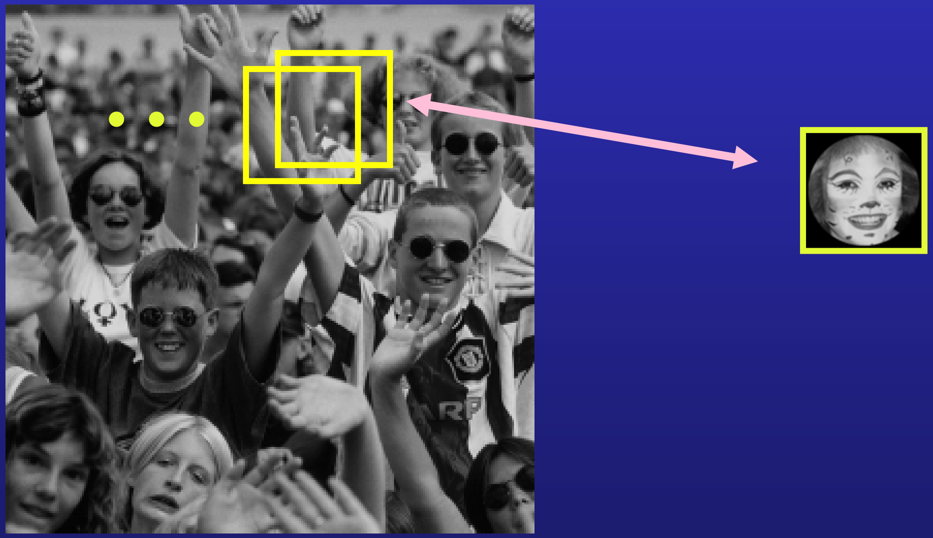

FIND THE FACE
File
Edit
Tools
Window
Help
start
next
close
info
result (shown in <– pict.)
unclss pix.
1
%
clutter pix.
99
%
dist calcs
16


FIND THE FACE
File Edit Tools Window Help
start
next
close
info
50
100
150
200
250
50
100
150
200
250
result (shown in <— pict.)
unclss pix.
0
%
clutter pix.
100
%
dist calcs
32

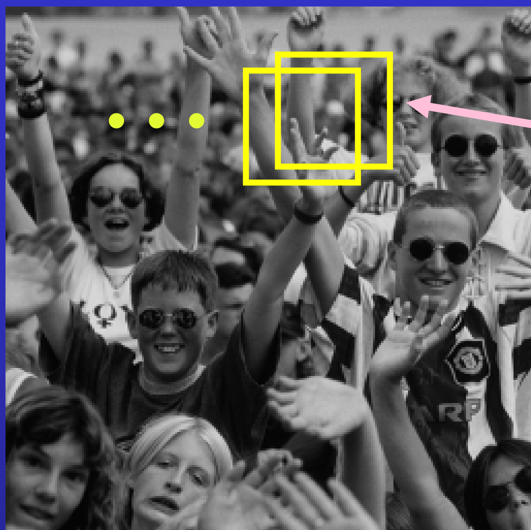
Fast Search in Group Orbit: Conclusions

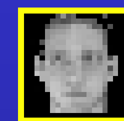
- Observation 1: Orbit distance is a metric when the point distance is transformation invariant.
- Observation 2: Fast search in orbit distance space can be applied using recursive orbit decomposition.
- Distant patterns are rejected fast.
- **Important**: Can be applied to any metric distance d(Q,P).

Efficient Search in the Spatial Domain

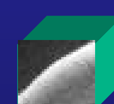


The Euclidean Distance





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



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

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

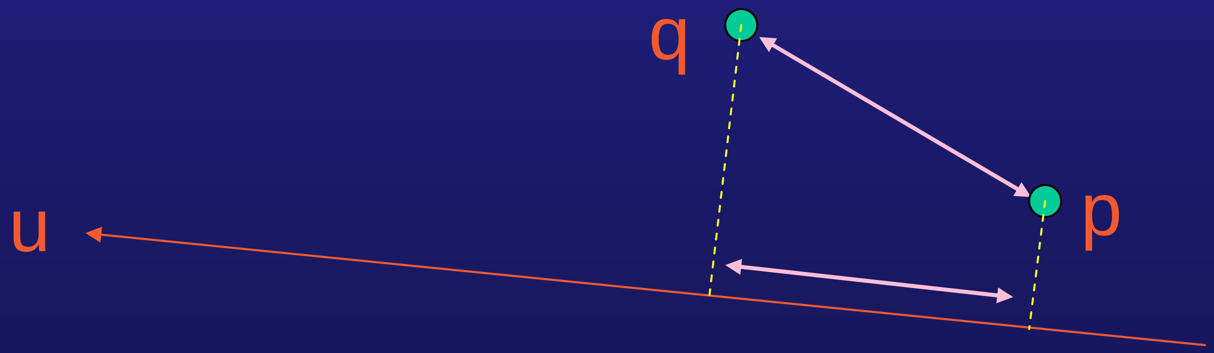
Norm Distance in Sub-space

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

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

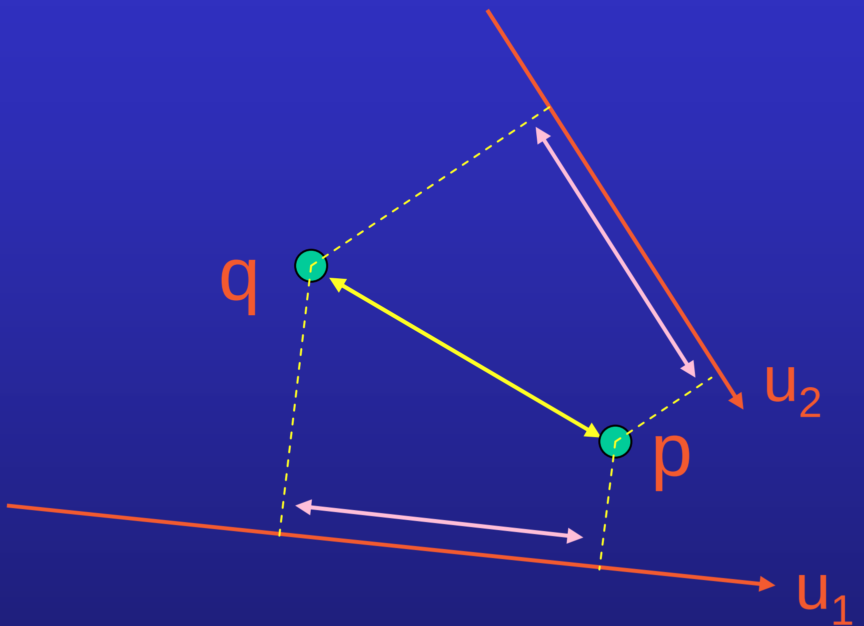
- If p and q were projected onto a kernel u, it follows from the Cauchy-Schwarz Inequality:

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



Distance Measure in Sub-space (Cont.)

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



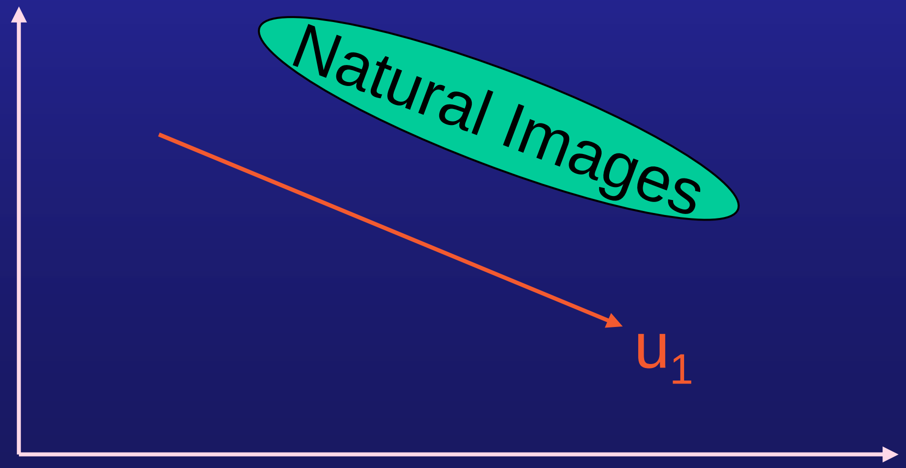
It can be shown that:

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

How can we Expedite the Distance Calculations?

Two necessary requirements:

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



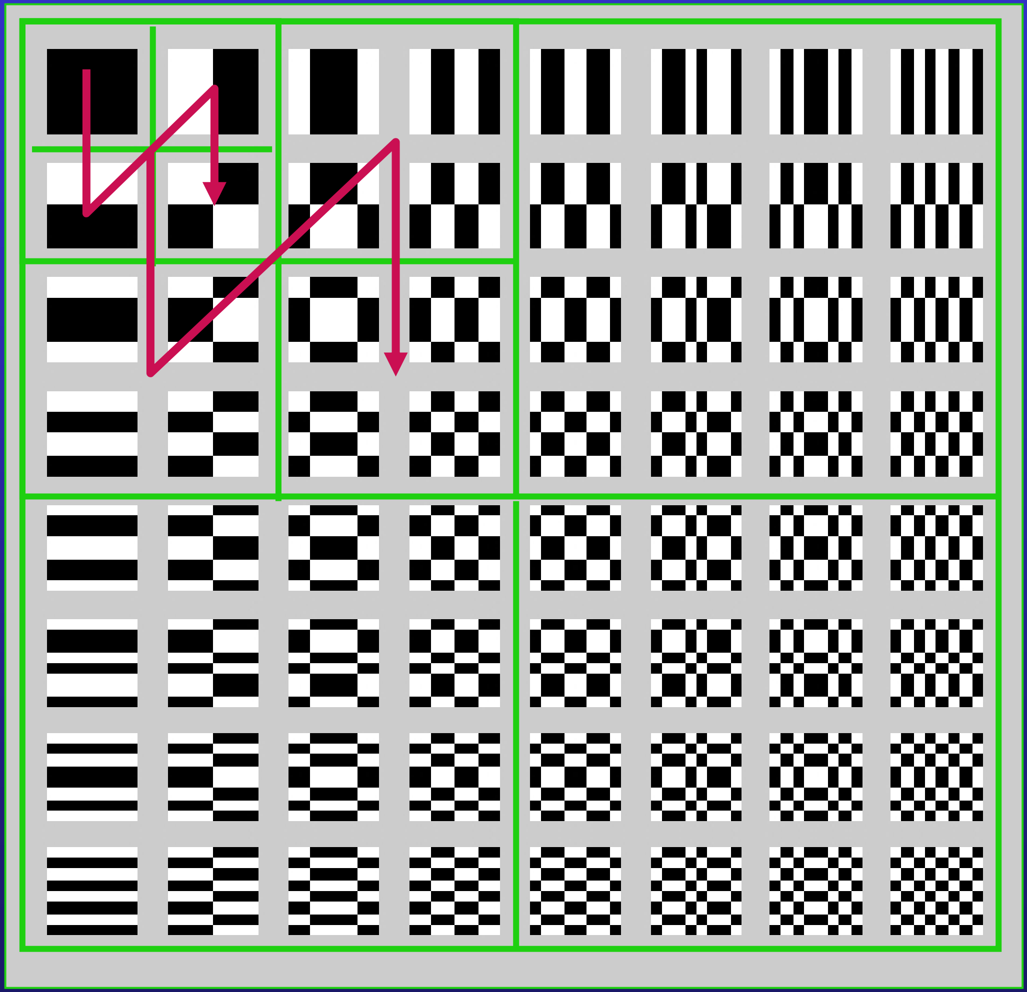
Projecting Kernels: Walsh-Hadamard

Following the above requirement we use the kxk

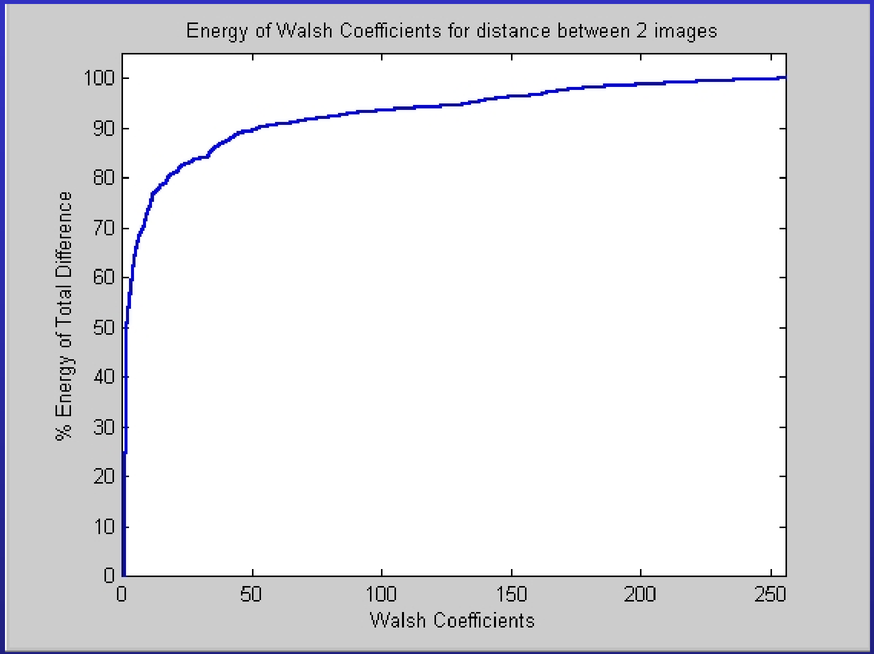
Walsh-Hadamard kernels

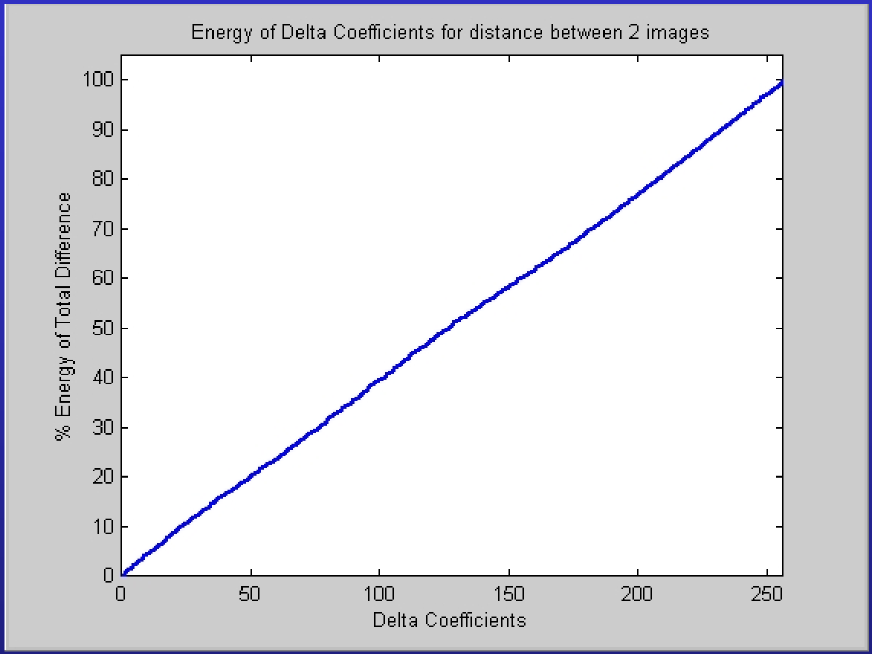
- Each window in a natural image is closely spanned by the first few kernel vectors.
- Can be applied very fast in a recursive manner.

The Walsh-Hadamard Kernels:



Walsh-Hadamard v.s. Standard Basis:

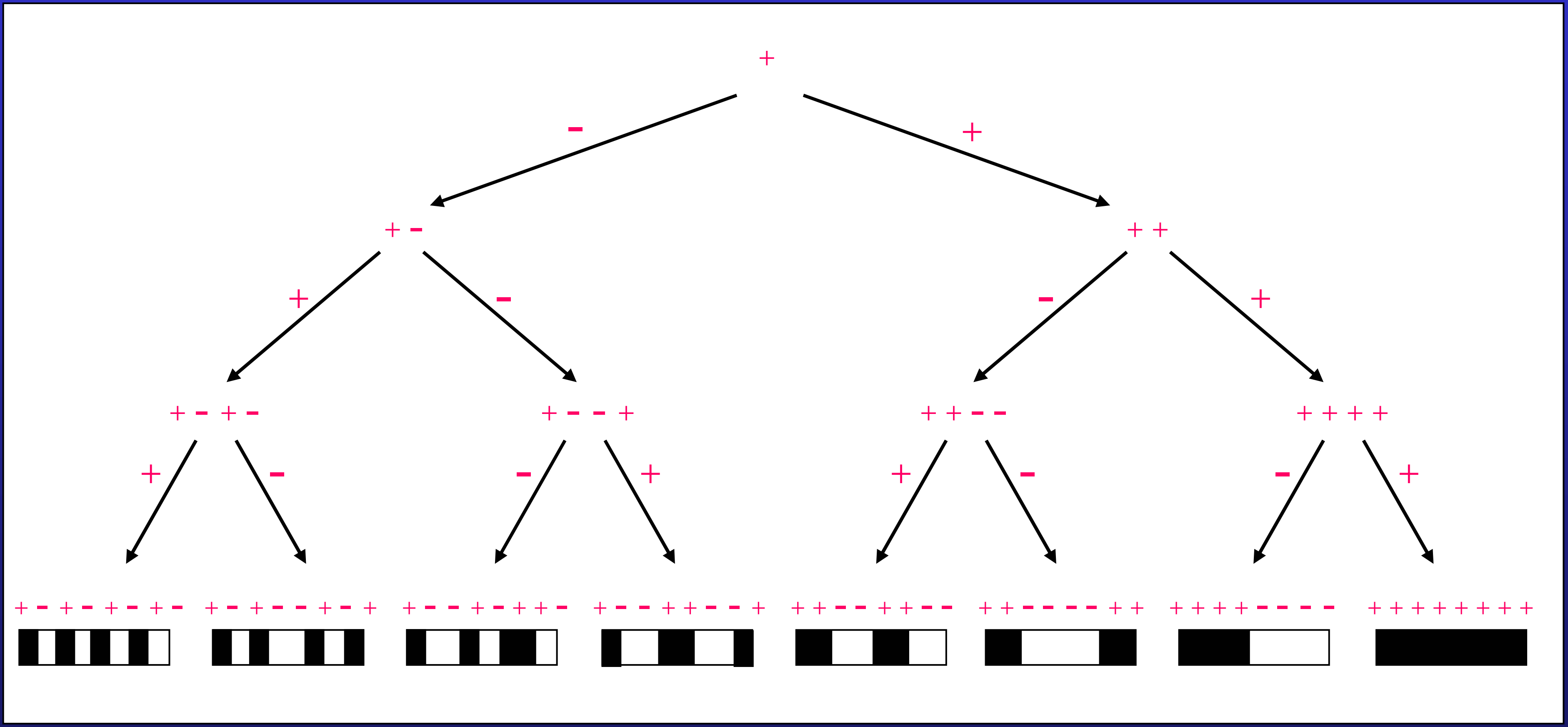




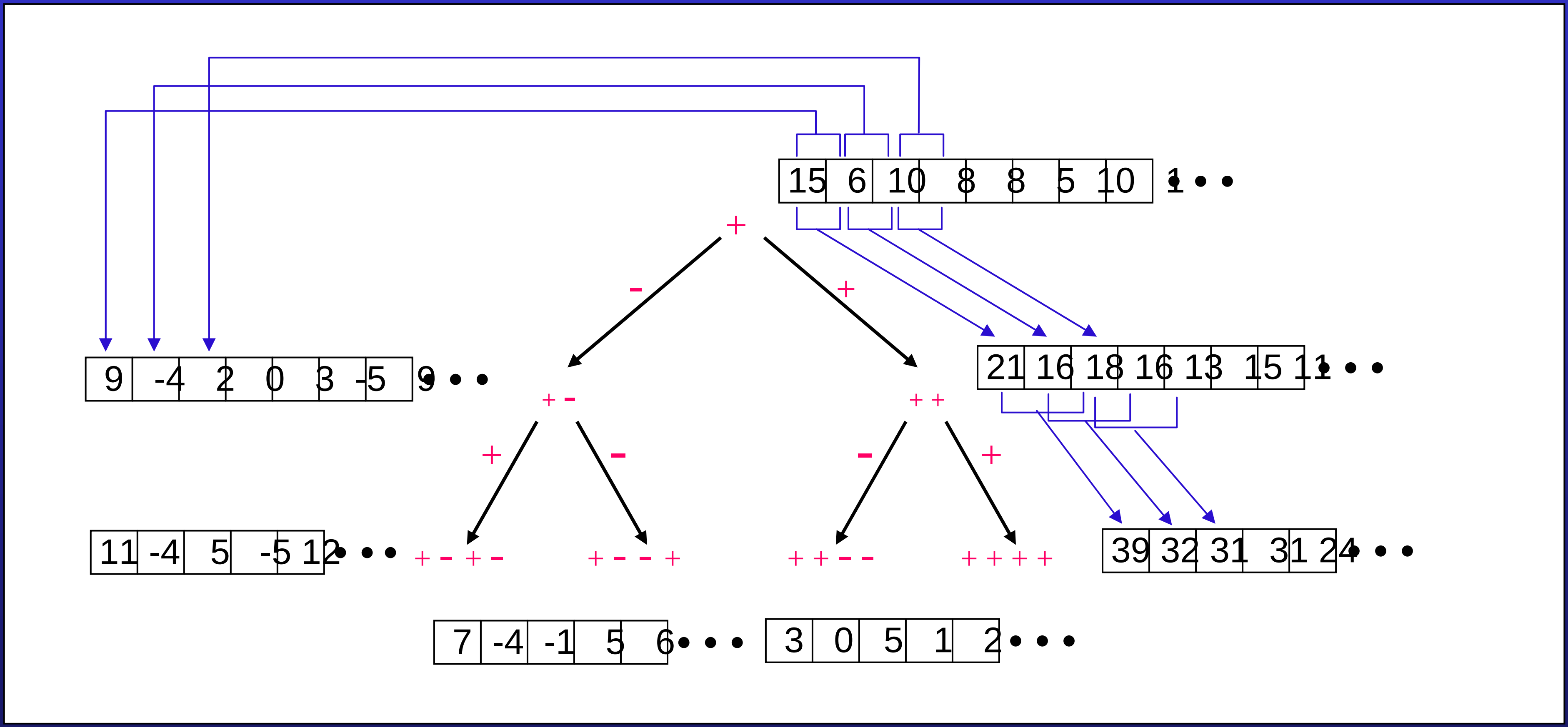
The lower bound for distance value in % v.s. number of Walsh-Hadamard projections,
Averaged over 100 pattern-image pairs of size 256x256 .

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)

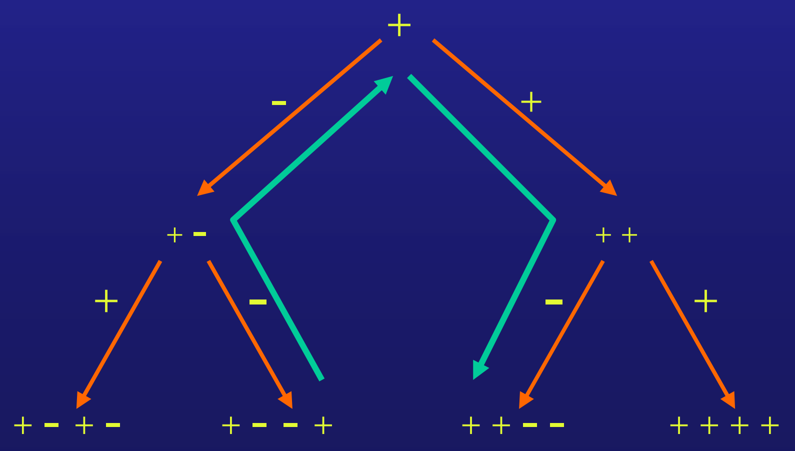


The Walsh-Hadamard Tree - Example



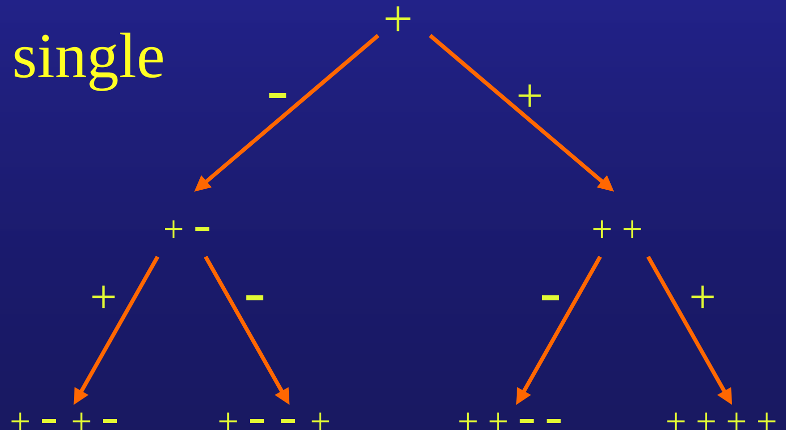
Properties:

- Descending from a node to its child requires one addition operation per pixel (convolution).
- A projection of the entire image onto one kernel is performed in a top-down traversal.
- A projection of a particular window in the image onto one kernel is performed in a bottom-up traversal.
- All operations are performed in integers.



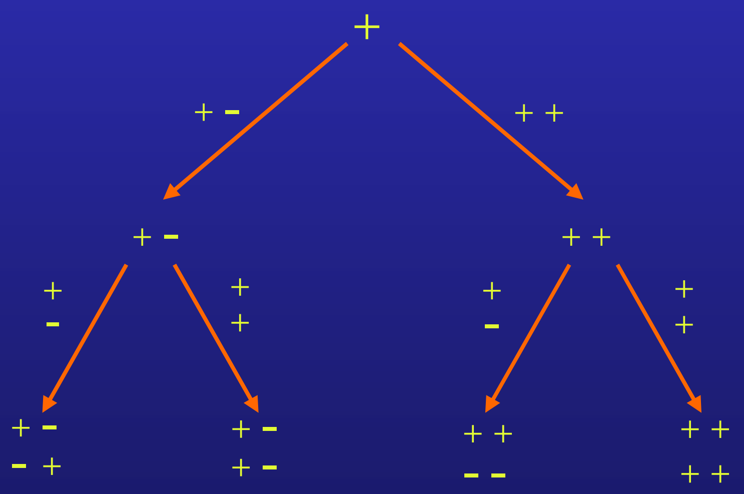
Complexity (1D):

- Projecting **all** windows in the image onto a single kernel requires *log k* additions per pixel.
- Projecting **all** windows in the image onto $l<k$ kernels requires *m* additions per pixel, where *m* is the number of nodes preceding the *l* leaf.
- Projecting **all** windows in the image onto *k* kernels requires *2k* additions per pixel.
- Projecting a **single** window onto a single kernel requires k-1 additions.



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

Pattern Matching algorithm

- 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.

Pattern Matching algorithm - Complexity

All windows are projected onto the first kernel :

$2logk$ ops/pixel

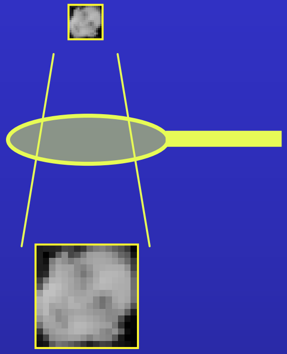
Only a few windows are further projected using ~2k operations per active window :

ε ops/pixel

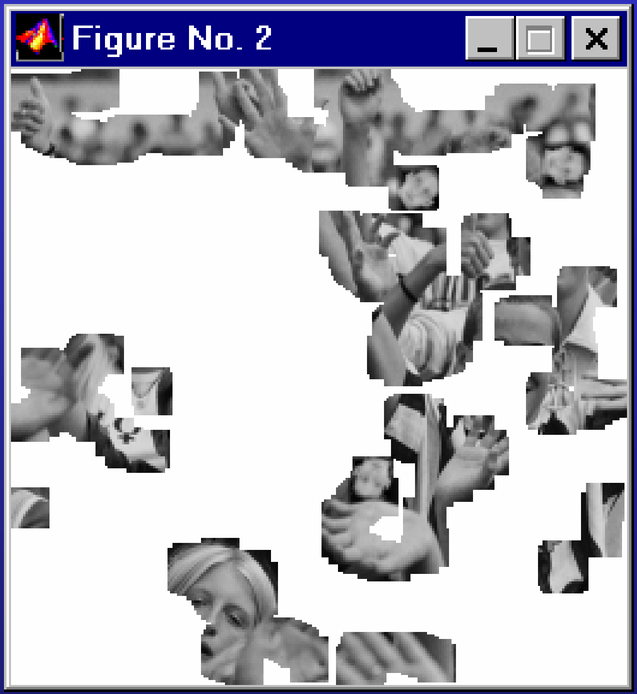
Total : $\boxed{2logk + \varepsilon \text{ ops/pixel}}$

Example:

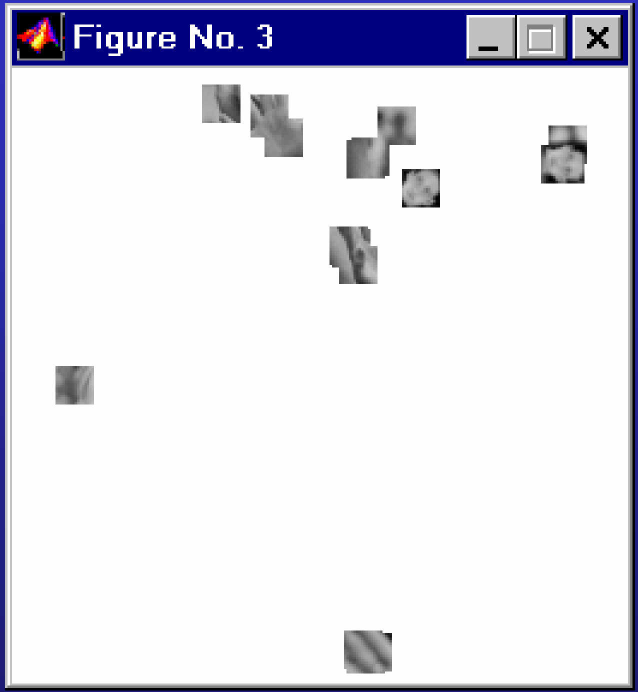
Initial Image: 65536 candidates



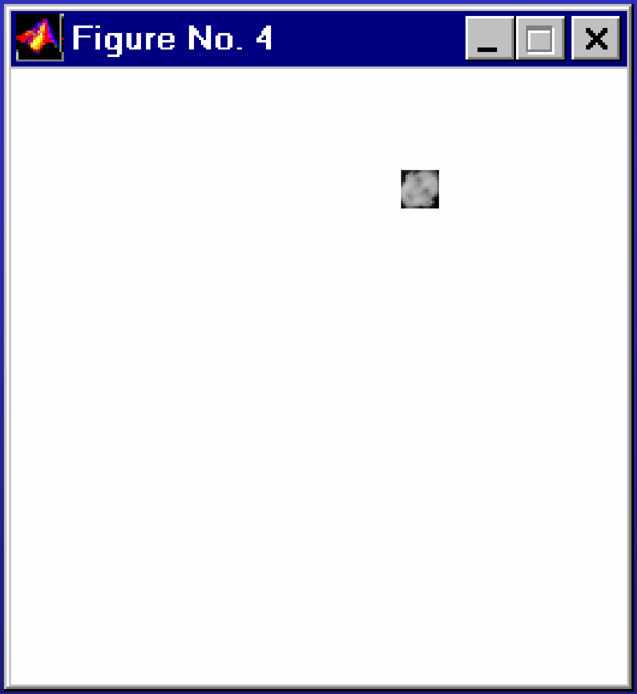
Sought Pattern



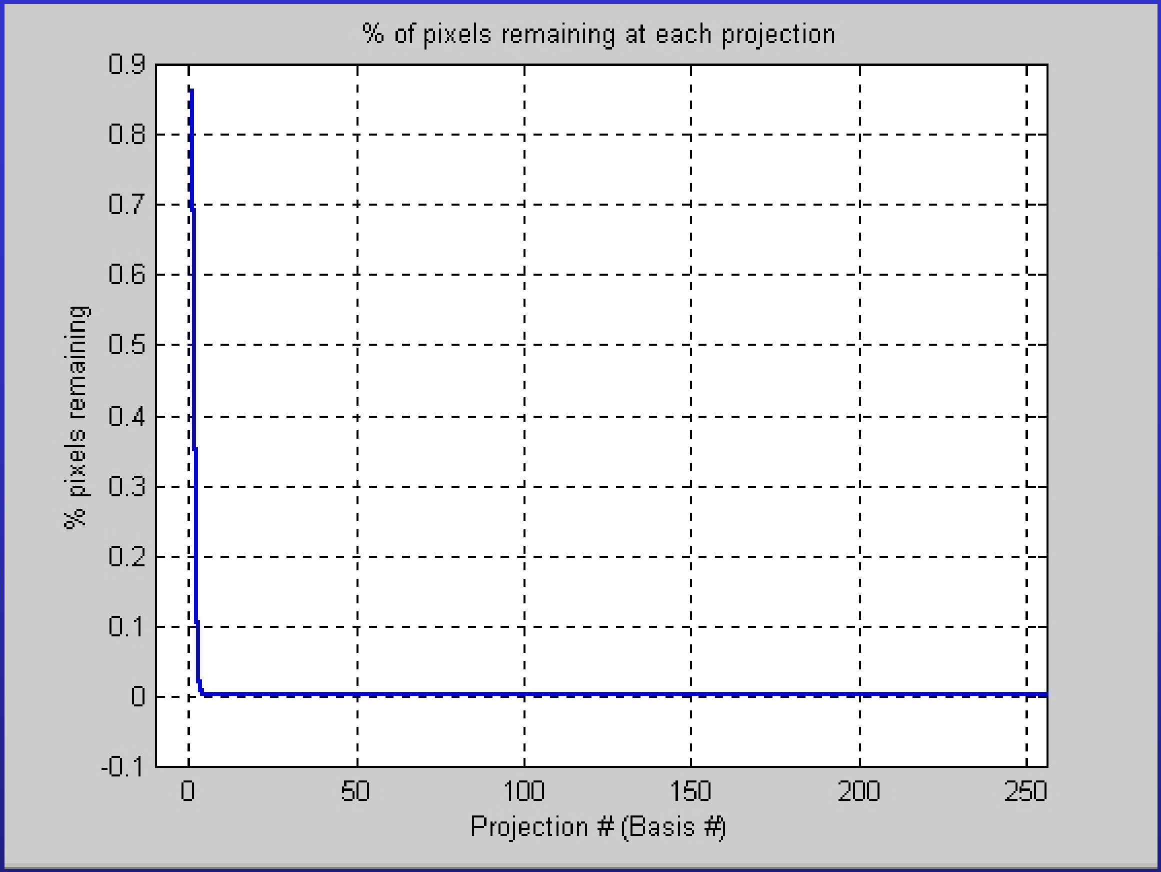
After the 1st projection: 563 candidates



After the 2nd projection: 16 candidates

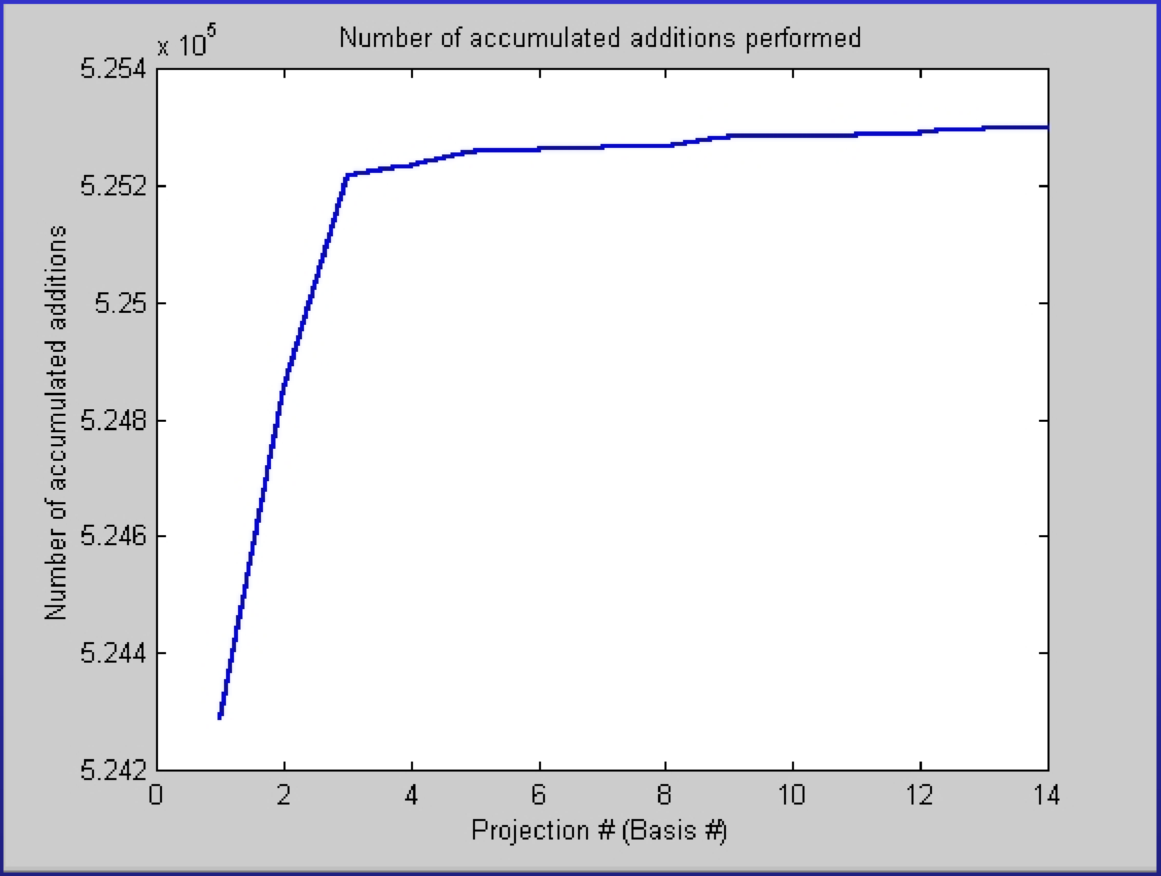


After the 3rd projection: 1 candidate



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

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



Accumulated number of additions after each projection averaged over 100 pattern-image pairs.
Image size = 256x256, pattern size = 16x16.

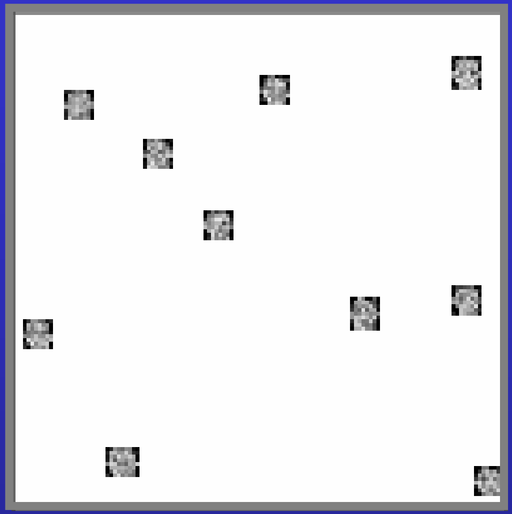
Average Number of operations per pixel: 8.0154

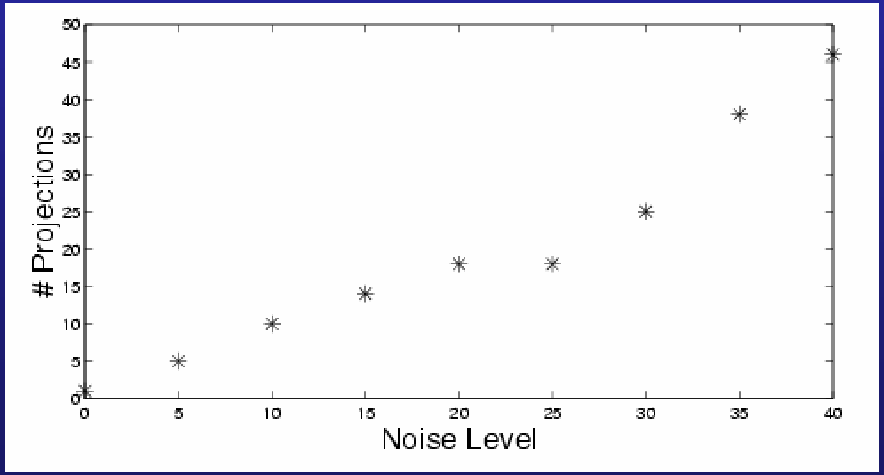
Example with Noise

Original

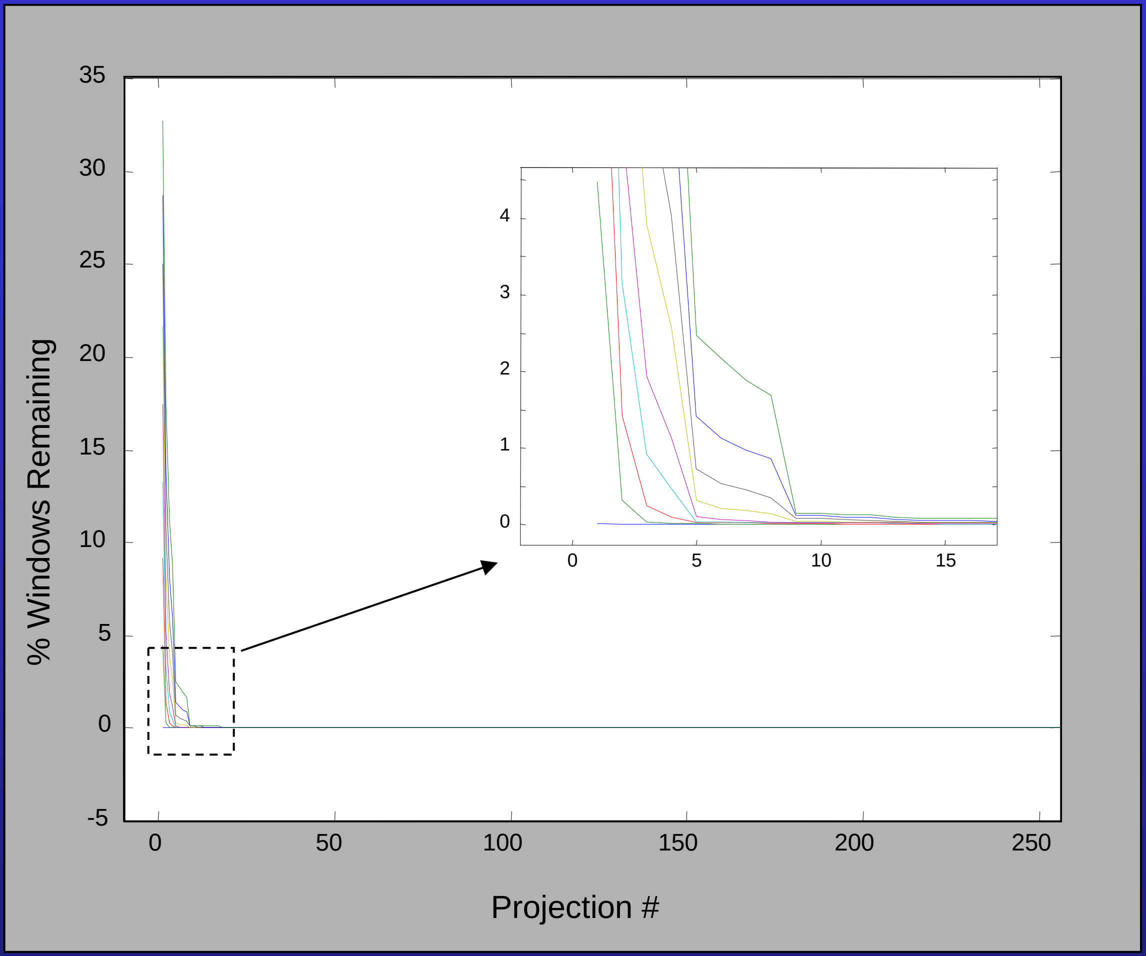
Noise Level = 40

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.

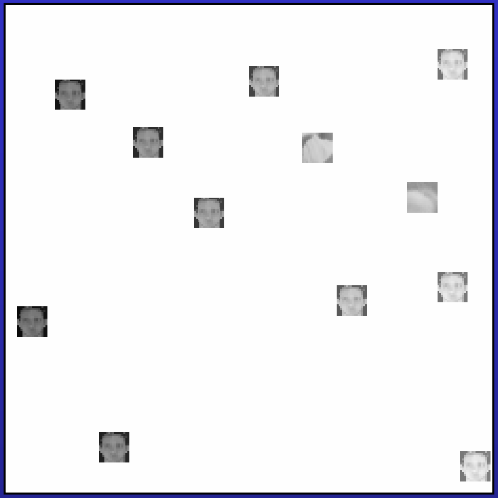
Image size = 256x256, pattern size = 16x16.

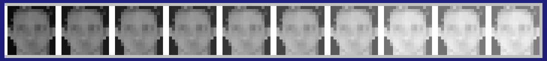
DC-invariant Pattern Matching

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:

- Walsh-Hadamard per window can be applied very fast.
- Projections are performed with additions/subtractions only (no multiplications).
- Integer operations (3 times faster for additions).
- Fast rejection of windows.
- Possible to perform pattern matching at video rate.
- Extensions:
  - DC invariant pattern matching.
  - Other norms.
  - Multi size pattern matching.

Limitations:

- $2n^2 \log k$ memory size.
- Pattern size must be 2^m.
- Limited to normed distance metrics.

Conclusion

Pattern Detection using 2 complementary processes:

1. Reduce search in Transformation Domain.
2. Reduce search in Spatial Domain.

Processes are based on rejection schemes, and are restricted to a specific domain.

The two processes can be combined into a single, highly efficient, search process.

--- END ---