Orthogonal moments

$\{p_{pq}(x,y)\}$ **- set of orthogonal polynomials**

$$v_{pq} = \iint_{\Omega} p_{pq}(x,y) f(x,y) \mathrm{d}x$$

Motivation for using OG moments

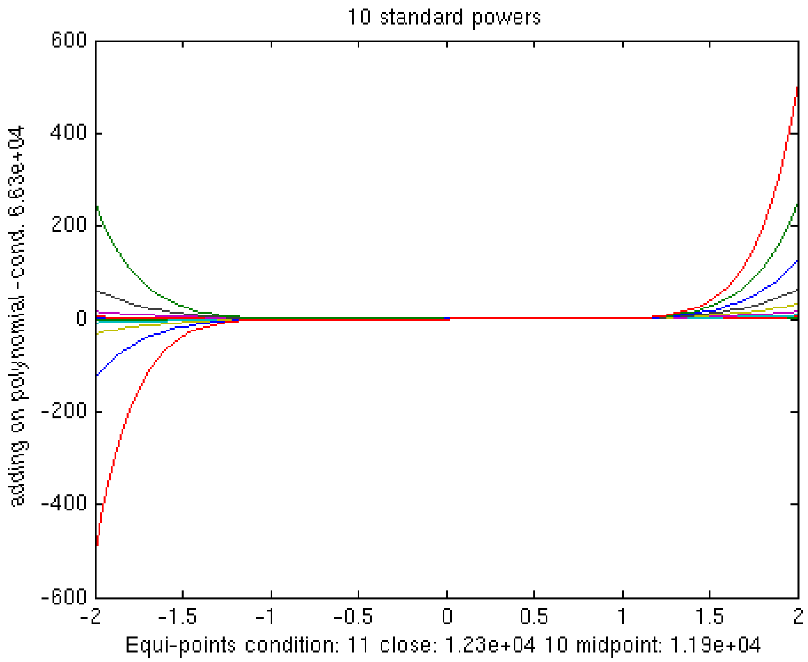
- **Stable calculation by recurrent relations**
- **Easier and stable image reconstruction**

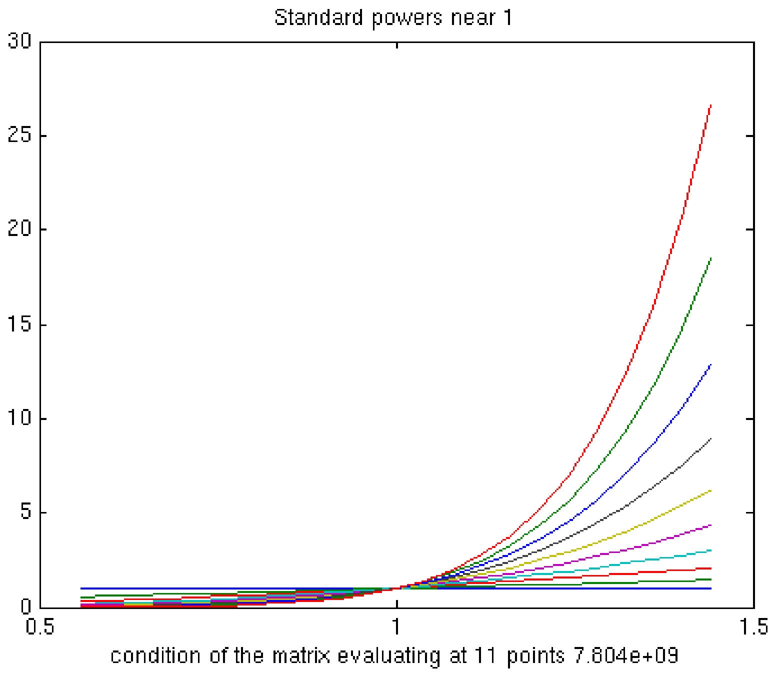
Numerical stability

How to avoid numerical problems with high dynamic range of geometric moments?

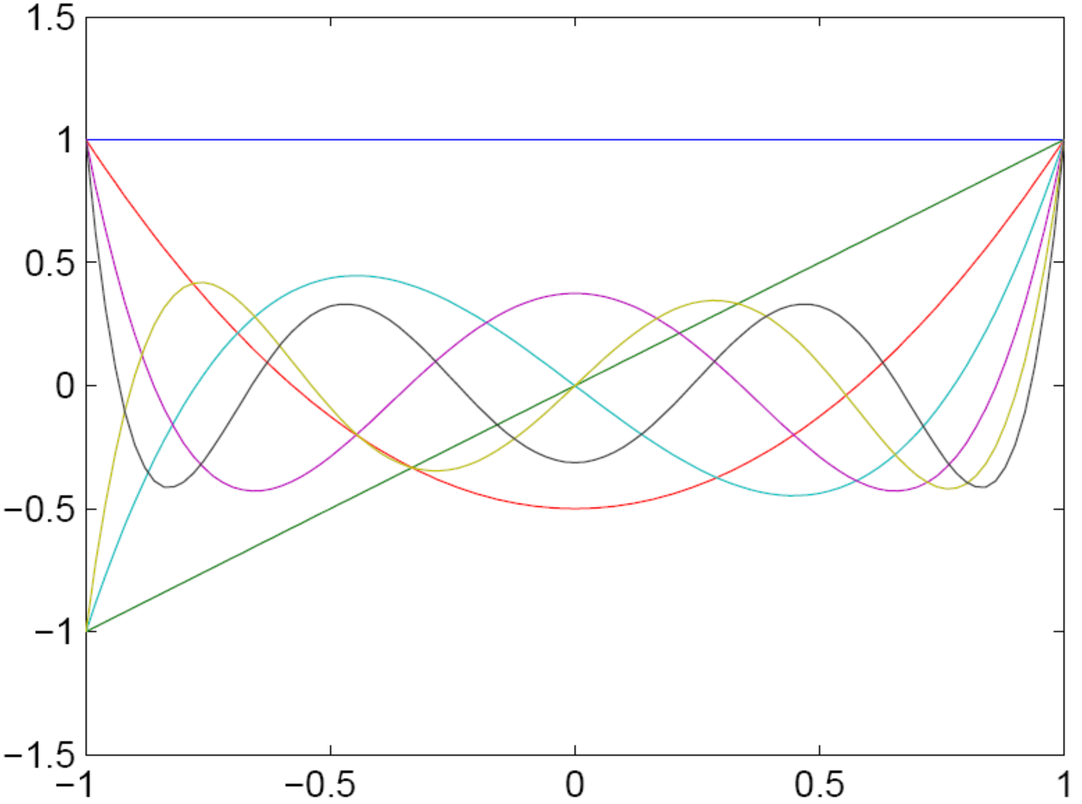
$$p_{j+1}(x) = xp_j(x) - \beta_j p_{j-1}(x)$$

Standard powers





Orthogonal polynomials



Calculation using recurrent relations

Two kinds of orthogonality

- Moments (polynomials) orthogonal on a unit square
- Moments (polynomials) orthogonal on a unit disk

Moments orthogonal on a square

$$v_{pq} = n_p n_q \iint\limits_{\Omega} p_p(x) p_q(y) f(x,y)\, dx\, dy$$

$p_k(x)$ is a system of 1D orthogonal polynomials

Common 1D orthogonal polynomials

- Legendre <-1,1>
- Chebyshev <-1,1>
- Gegenbauer <-1,1>
- Jacobi <-1,1> or <0,1>
- (generalized) Laguerre <0,∞)
- Hermite (-∞,∞)

Legendre polynomials

Definition

$$P_n(x) = \frac{1}{2^n n!} \frac{\mathrm{d}^n}{\mathrm{d}x^n} (x^2 - 1)^n$$

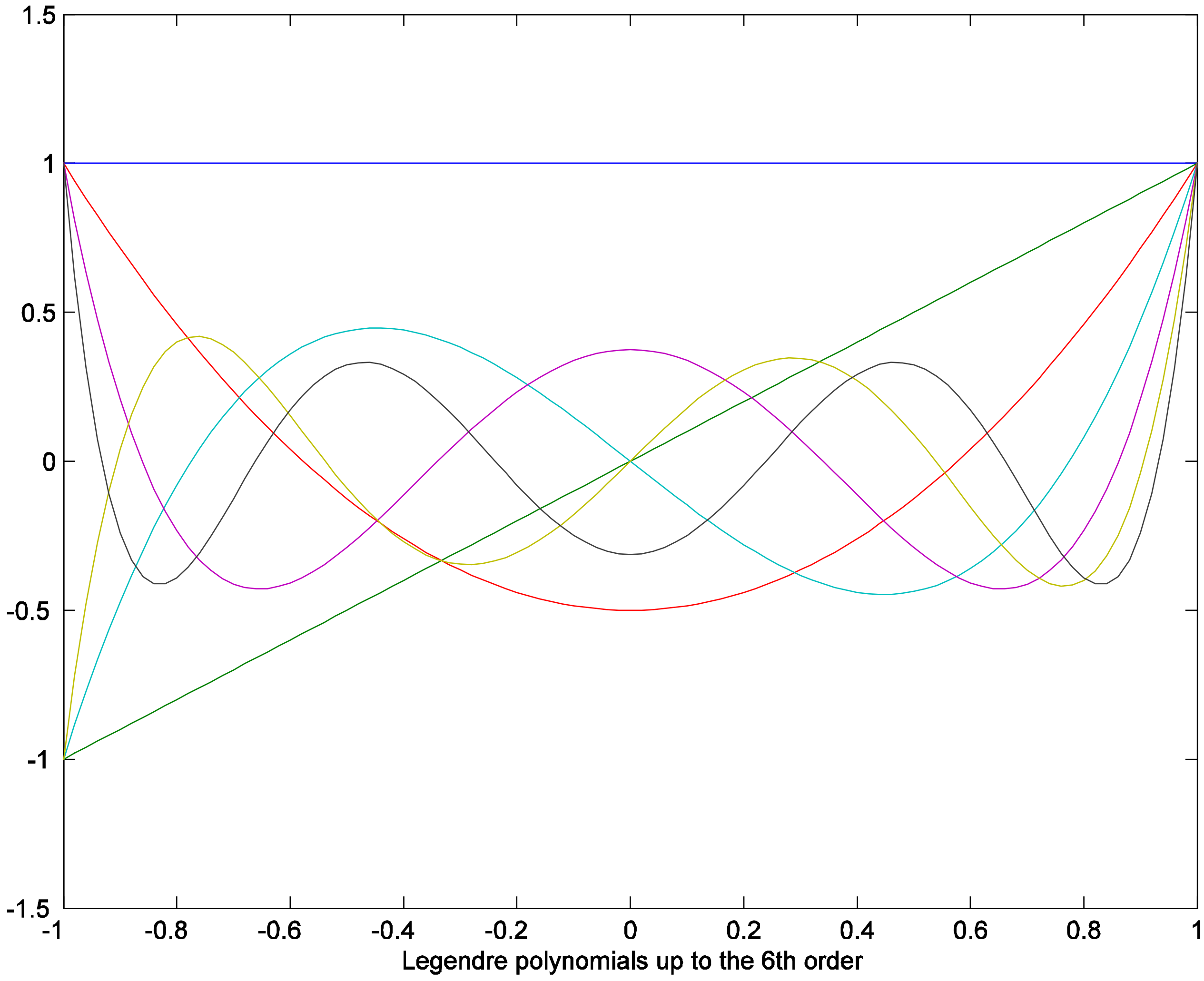
Orthogonality

$$\int_{-1}^{1} P_m(x) P_n(x) \ \mathrm{d}x = \frac{2}{2n+1} \delta_{mn}$$

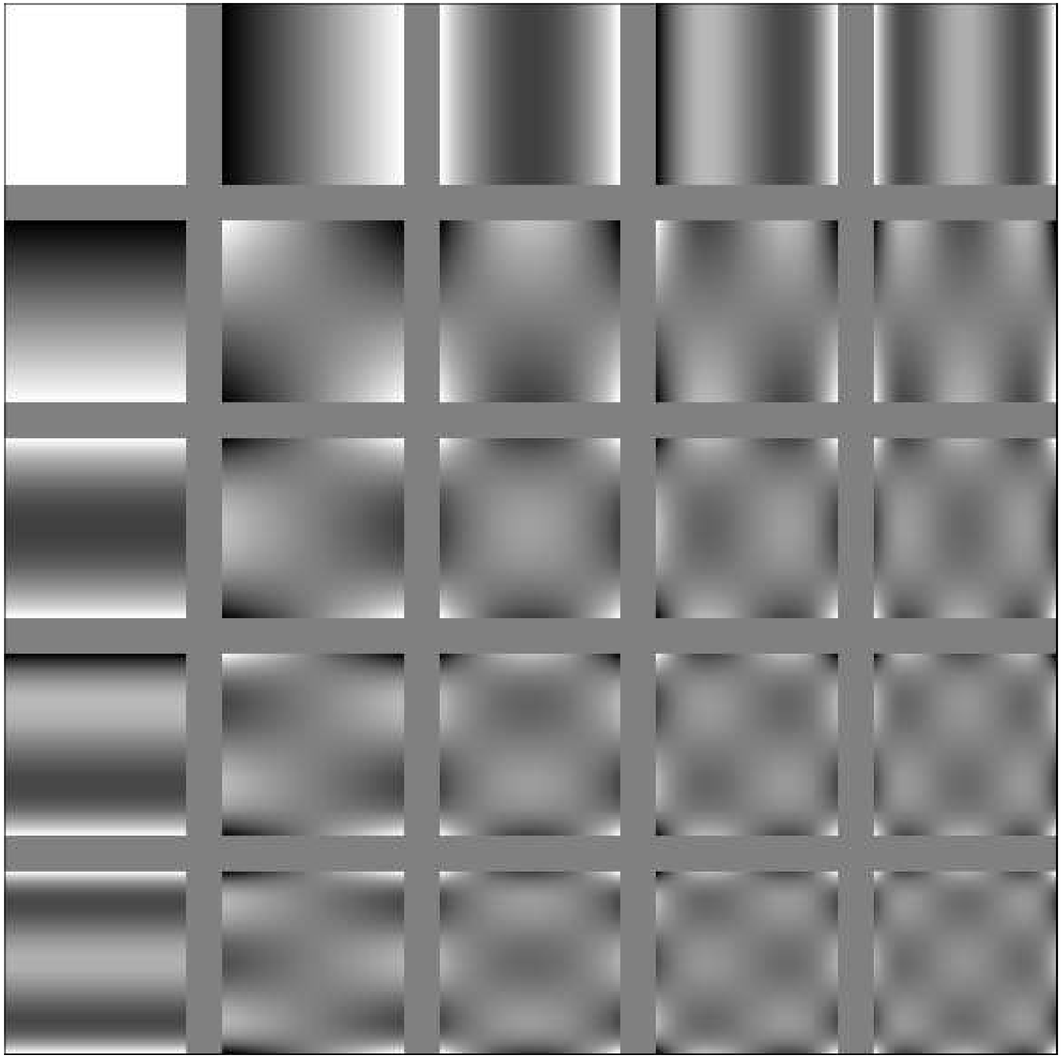
Legendre polynomials explicitly

$$
\begin{aligned}
P_0(x) &= 1, \\
P_1(x) &= x, \\
P_2(x) &= \tfrac{1}{2}(3x^2 - 1), \\
P_3(x) &= \tfrac{1}{2}(5x^3 - 3x), \\
P_4(x) &= \tfrac{1}{8}(35x^4 - 30x^2 + 3), \\
P_5(x) &= \tfrac{1}{8}(63x^5 - 70x^3 + 15x), \\
P_6(x) &= \tfrac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5).
\end{aligned}
$$

Legendre polynomials in 1D



Legendre polynomials in 2D



Legendre polynomials

Recurrent relation

$$
\begin{aligned}
P_0(x) &= 1, \\
P_1(x) &= x, \\
P_{n+1}(x) &= \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x).
\end{aligned}
$$

Legendre moments

$$\lambda_{mn} = \frac{(2m+1)(2n+1)}{4} \int_{-1}^{1} \int_{-1}^{1} P_m(x) P_n(y) f(x,y) \, \mathrm{d}x \, \mathrm{d}y$$

$$\lambda_{mn} = \frac{(2m+1)(2n+1)}{4} \sum_{p=0}^{m} \sum_{q=0}^{n} a_{mp} a_{nq} m_{pq}$$

Moments orthogonal on a disk

$$v_{pq} = n_{pq} \int_0^{2\pi} \int_0^1 R_{pq}(r) e^{-iq\varphi} f(r,\varphi) r \, \mathrm{d}r \, \mathrm{d}\varphi$$

Radial part $R_{pq}(r)$

Angular part $e^{-iq\varphi}$

Moments orthogonal on a disk

- Zernike
- Pseudo-Zernike
- Orthogonal Fourier-Mellin
- Jacobi-Fourier
- Chebyshev-Fourier
- Radial harmonic Fourier

Zernike polynomials

Definition

$$A_{n\ell} = \frac{n+1}{\pi} \int_0^{2\pi} \int_0^1 V_{n\ell}^*(r,\varphi) f(r,\varphi) r \; \mathrm{d}r \, \mathrm{d}\varphi$$

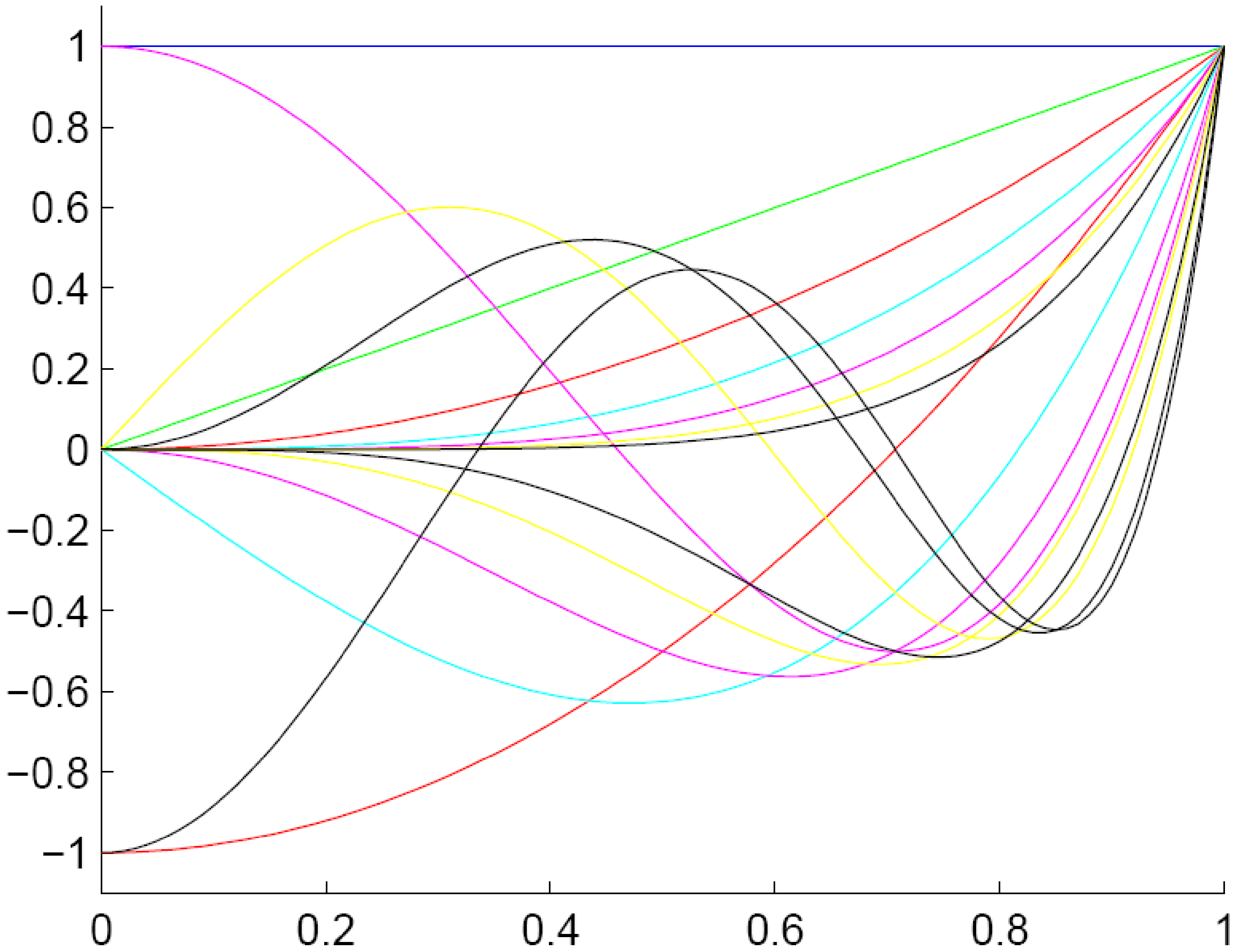
$$V_{n\ell}(r,\varphi) = R_{n\ell}(r) \; e^{i\ell\varphi}$$

$$R_{n\ell}(r) = \sum_{s=0}^{(n-|\ell|)/2} (-1)^s \frac{(n-s)!}{s!((n+|\ell|)/2-s)!((n-|\ell|)/2-s)!} r^{n-2s}$$

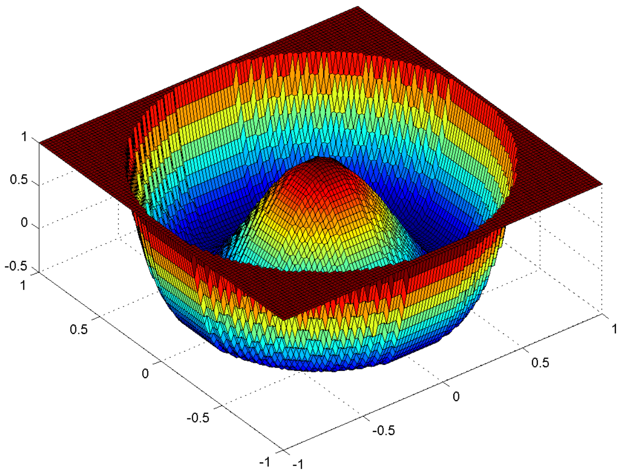
Orthogonality

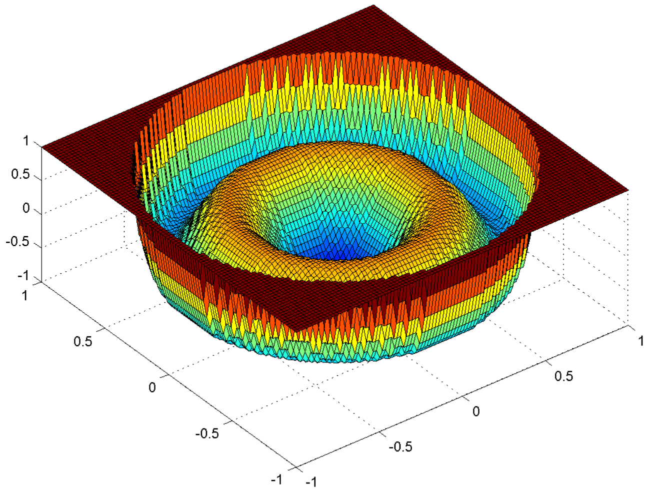
$$\int_0^{2\pi} \int_0^1 V_{n\ell}^*(r,\varphi) V_{mk}(r,\varphi) r \; \mathrm{d}r \; \mathrm{d}\varphi = \frac{\pi}{n+1} \; \delta_{mn} \delta_{k\ell}$$

Zernike polynomials – radial part in 1D

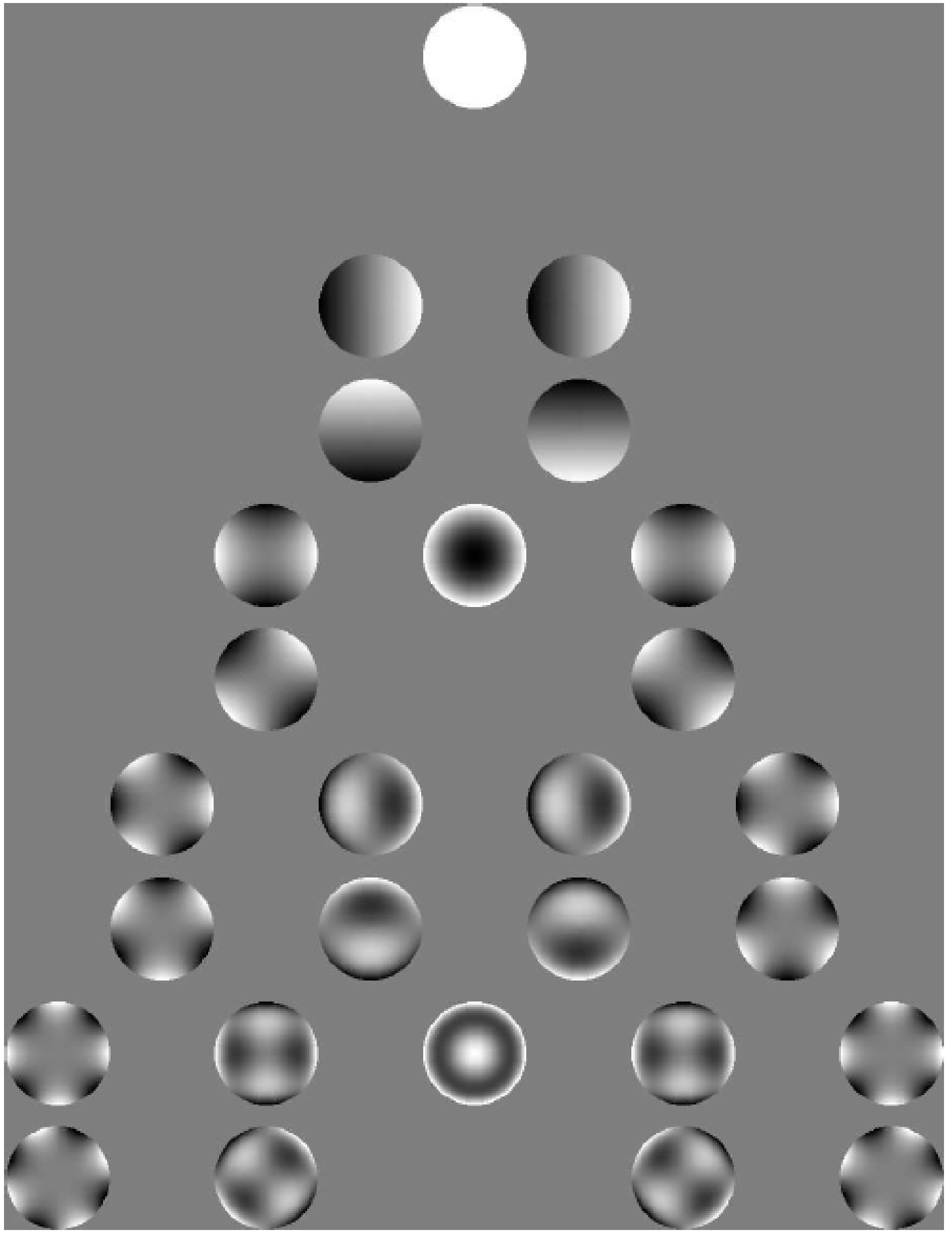


Zernike polynomials – radial part in 2D





Zernike polynomials



Zernike moments

$$A_{n\ell} = \frac{n+1}{\pi} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} V_{n\ell}^{*}(r, \varphi) f(x, y)$$

Mapping of Cartesian coordinates *x,y* to polar coordinates *r,φ*:

- Whole image is mapped inside the unit disk
- Translation and scaling invariance

Zernike moments

$$Z_{00} = \frac{1}{\pi} m_{00}$$

$$Z_{11} = \frac{2}{\pi} (m_{10} - i m_{01})$$

$$Z_{20} = \frac{6}{\pi} (m_{20} + m_{02}) - \frac{3}{\pi} m_{00}$$

Rotation property of Zernike moments

$$A'_{n\ell} = A_{n\ell} \; e^{-i\ell\,\theta}$$

The magnitude is preserved, the phase is shifted by $\ell\theta$.

Invariants are constructed by phase cancellation

Zernike rotation invariants

Phase cancellation by multiplication

$$\bar{Z}_{m\ell} = A_{m\ell}/(A_{m_r\ell_r})^{\ell/\ell_r}$$

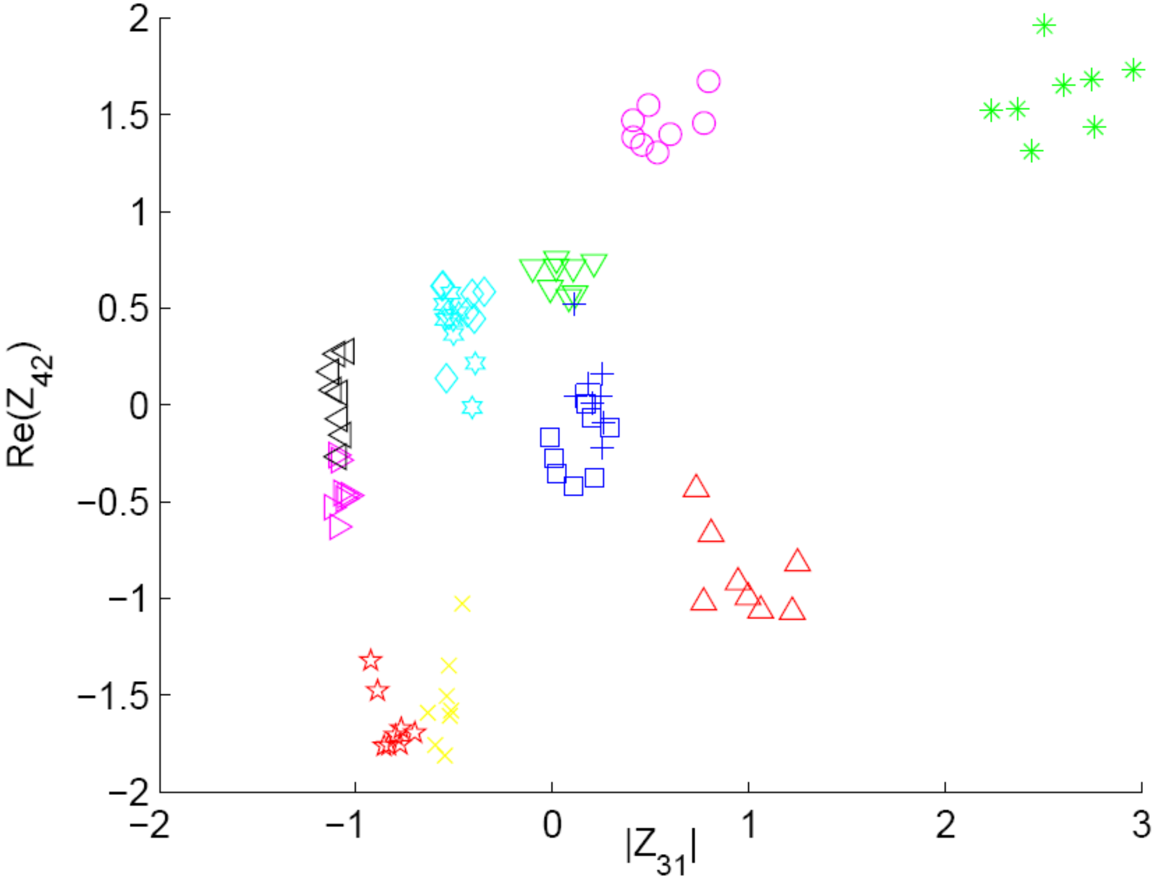
Normalization to rotation

$$\phi = \frac{1}{\ell_r}\arctan\left(\frac{\mathcal{I}m(A_{m_r\ell_r})}{\mathcal{R}e(A_{m_r\ell_r})}\right)$$

$$Z_{m\ell} = A_{m\ell}\; e^{-i\ell\,\phi}$$

Recognition by Zernike rotation invariants

Insufficient separability



Sufficient separability

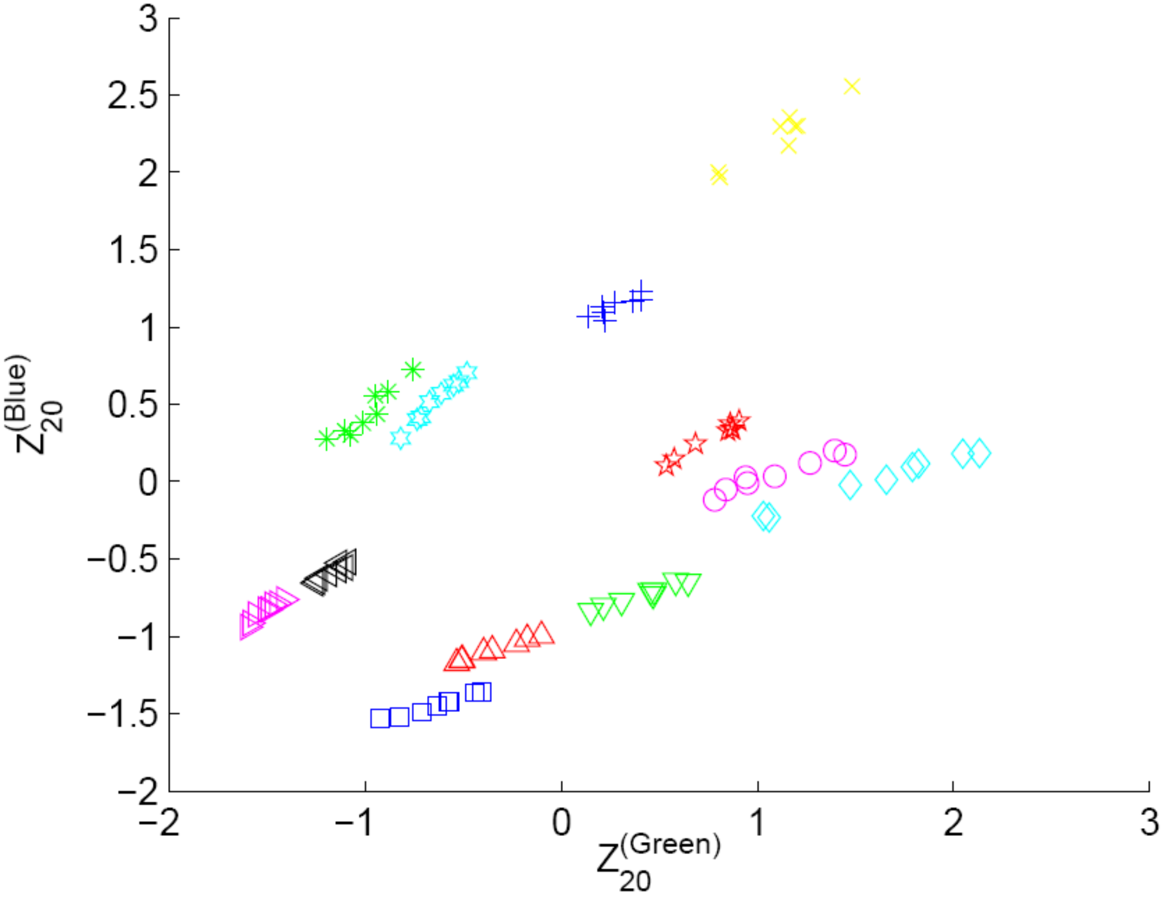


Image reconstruction

- Direct reconstruction from geometric moments

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y)$$

  - Solution of a system of equations
  - Works for very small images only
  - For larger images the system is ill-conditioned

Image reconstruction by direct computation

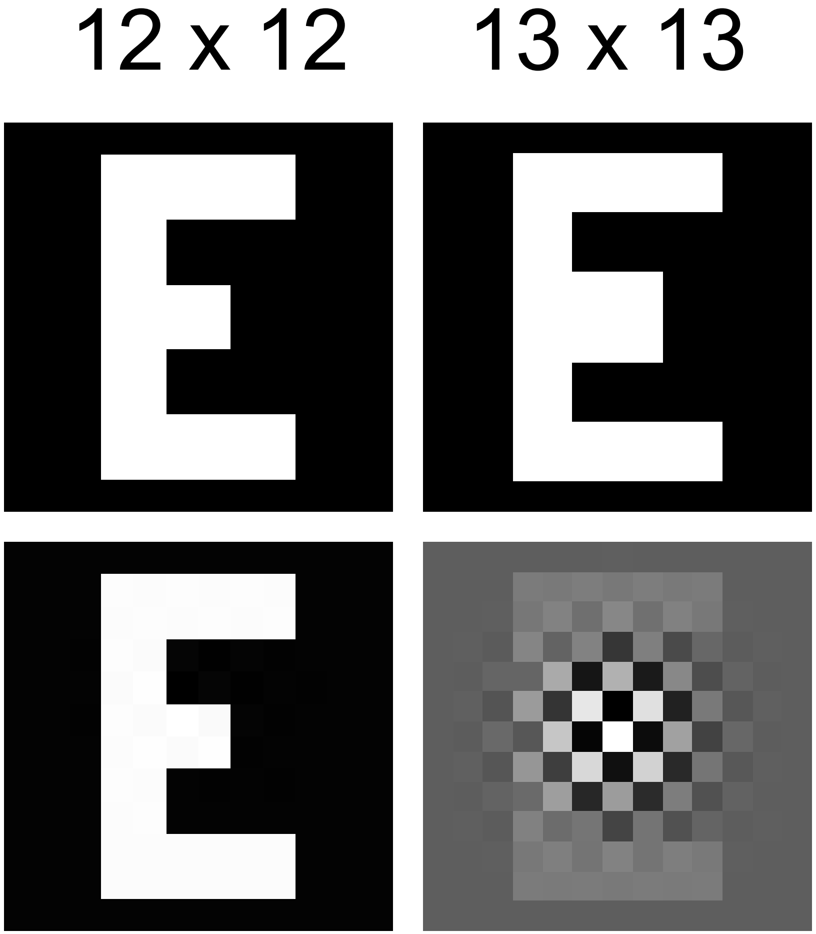


Image reconstruction

- Reconstruction from geometric moments via FT

$$F(u,v) = \sum_{p=0}^{\infty}\sum_{q=0}^{\infty} \frac{(-2\pi i)^{p+q}}{p!q!} \left(\frac{u}{M}\right)^p \left(\frac{v}{N}\right)^q m_{pq}$$

Image reconstruction via Fourier transform

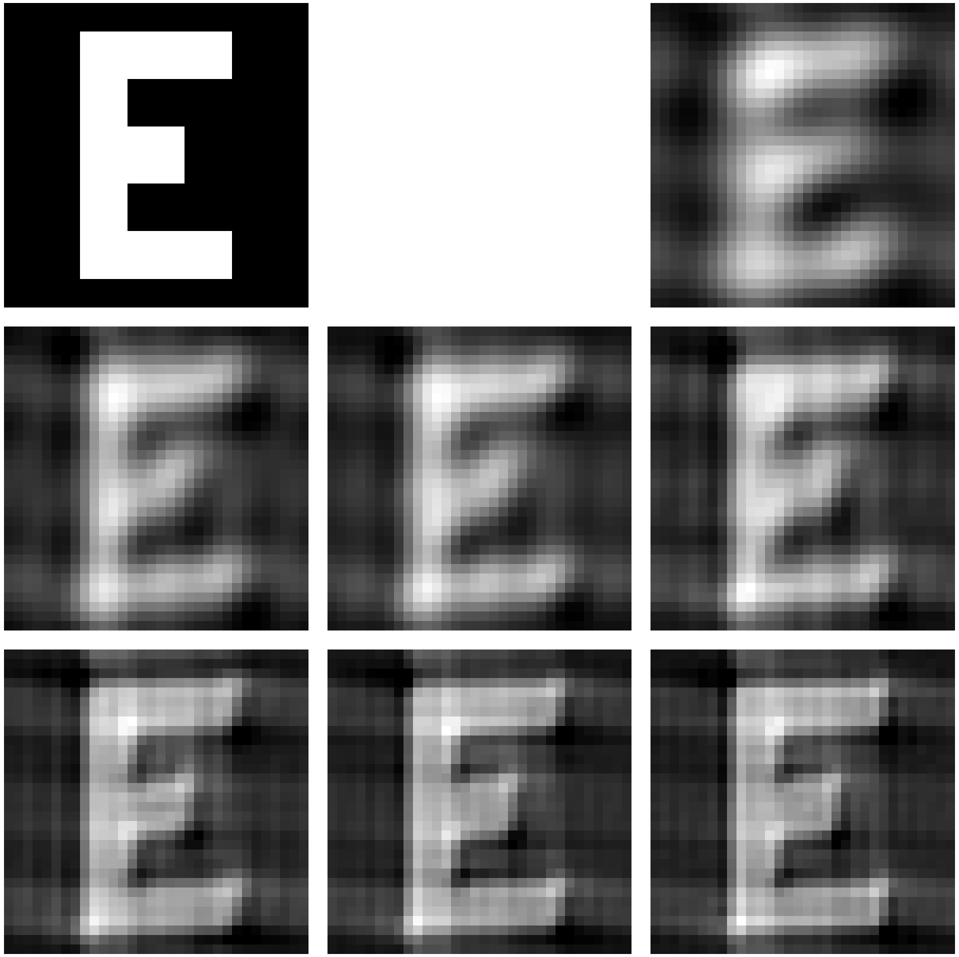


Image reconstruction

- Image reconstruction from OG moments on a square

$$f(x,y) = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} v_{pq} p_p(x) p_q(y)$$

- Image reconstruction from OG moments on a disk (Zernike)

$$f(x,y) = \sum_{n=0}^{\infty} \sum_{\ell=-n,-n+2,\ldots}^{n} A_{n\ell} V_{n\ell}(x,y)$$

Image reconstruction from Legendre moments

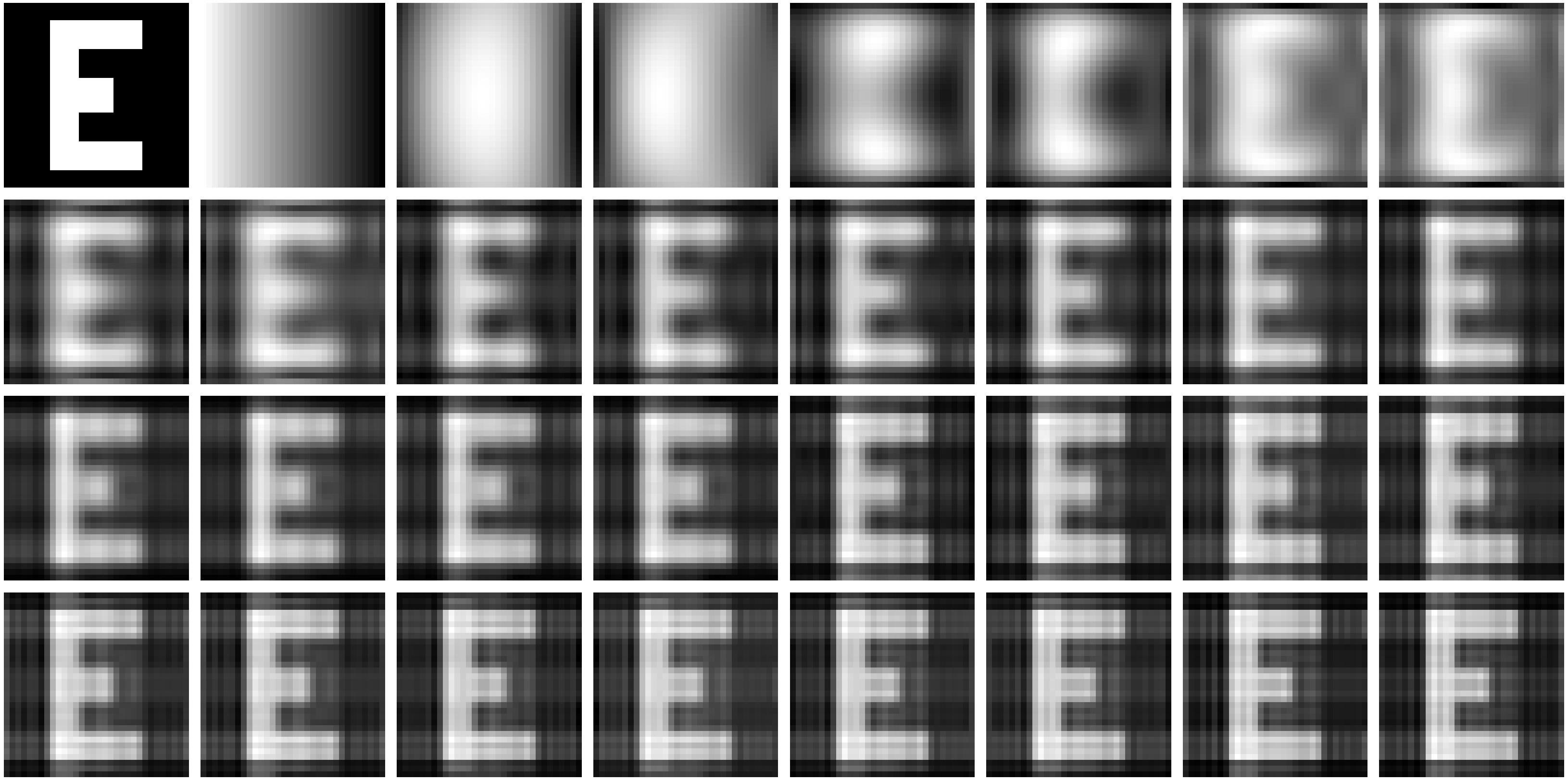
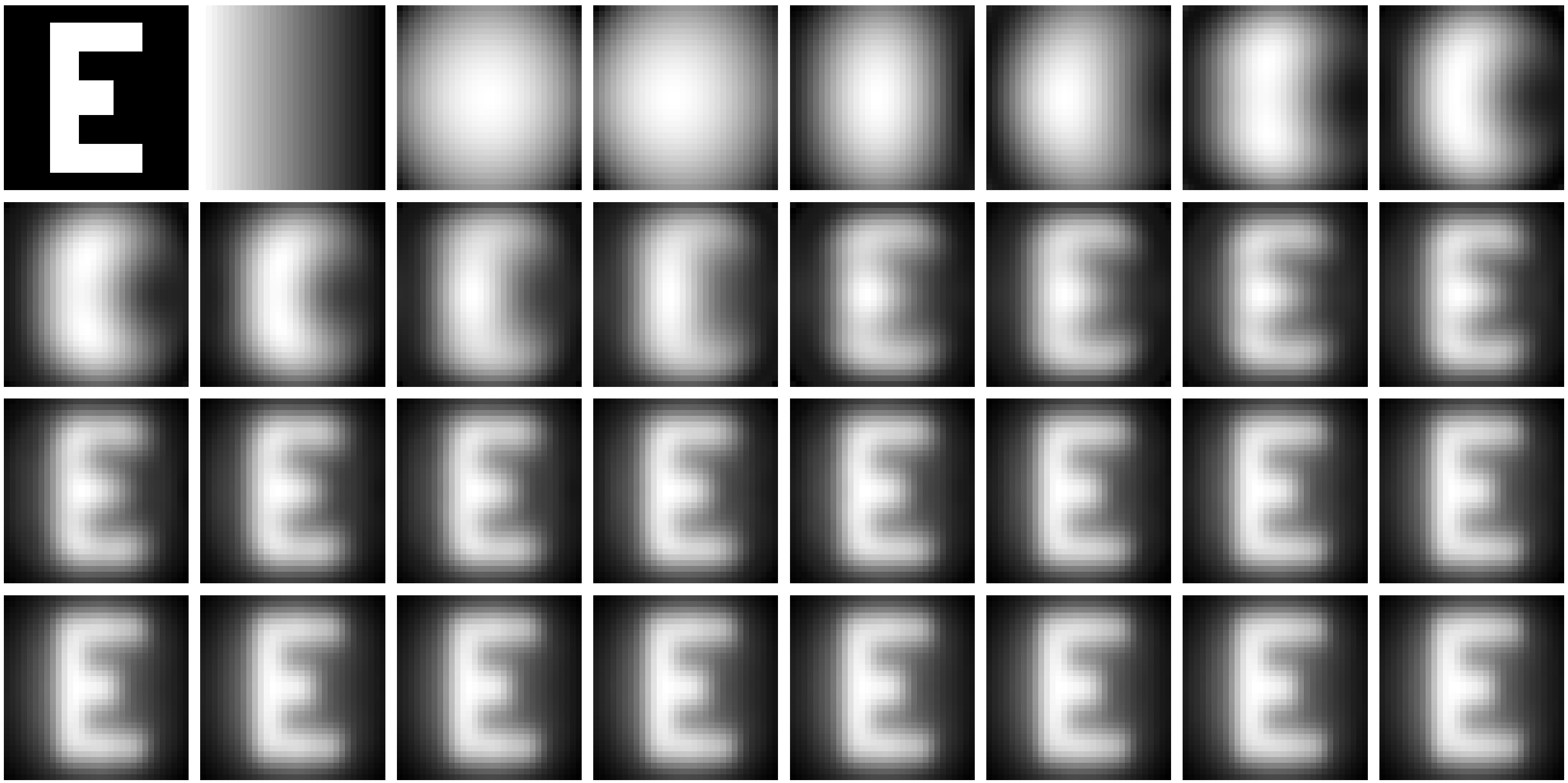
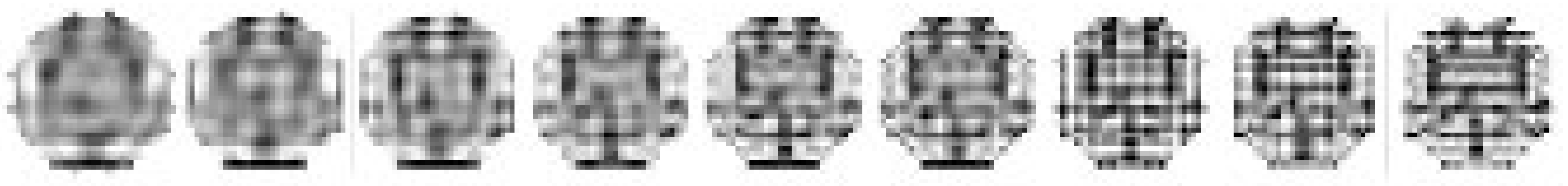


Image reconstruction from Zernike moments

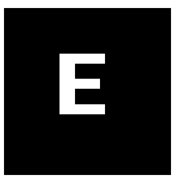


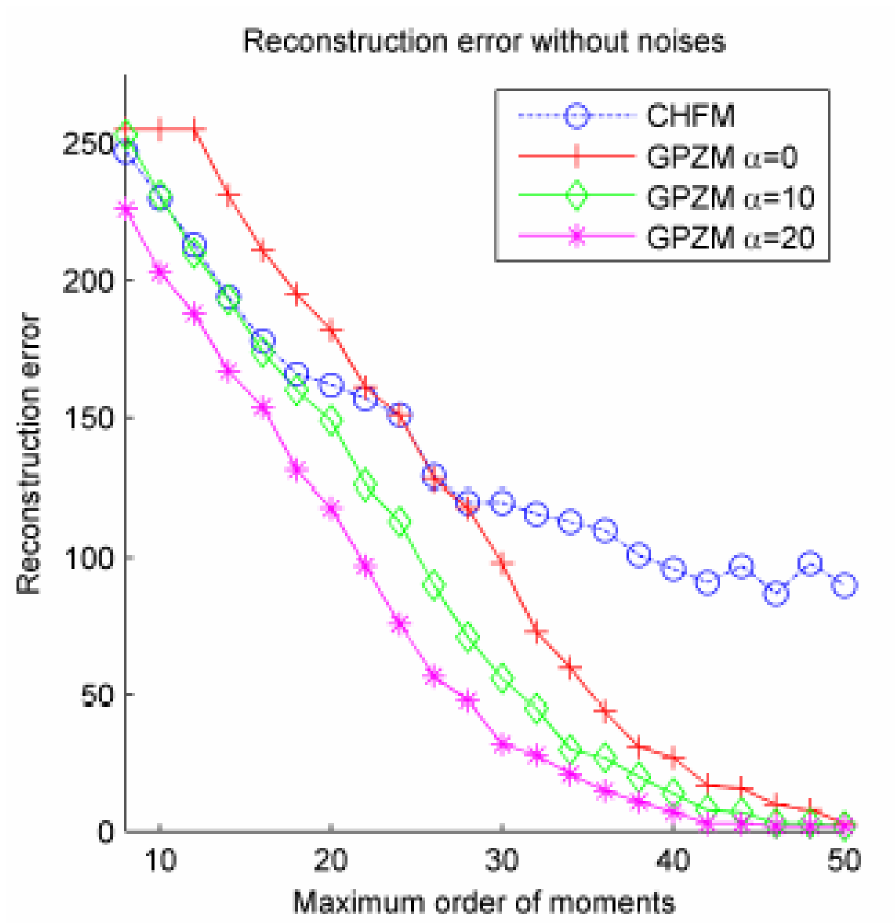
Better for polar raster

Image reconstruction from Zernike moments

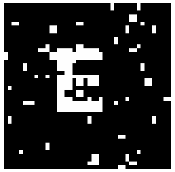


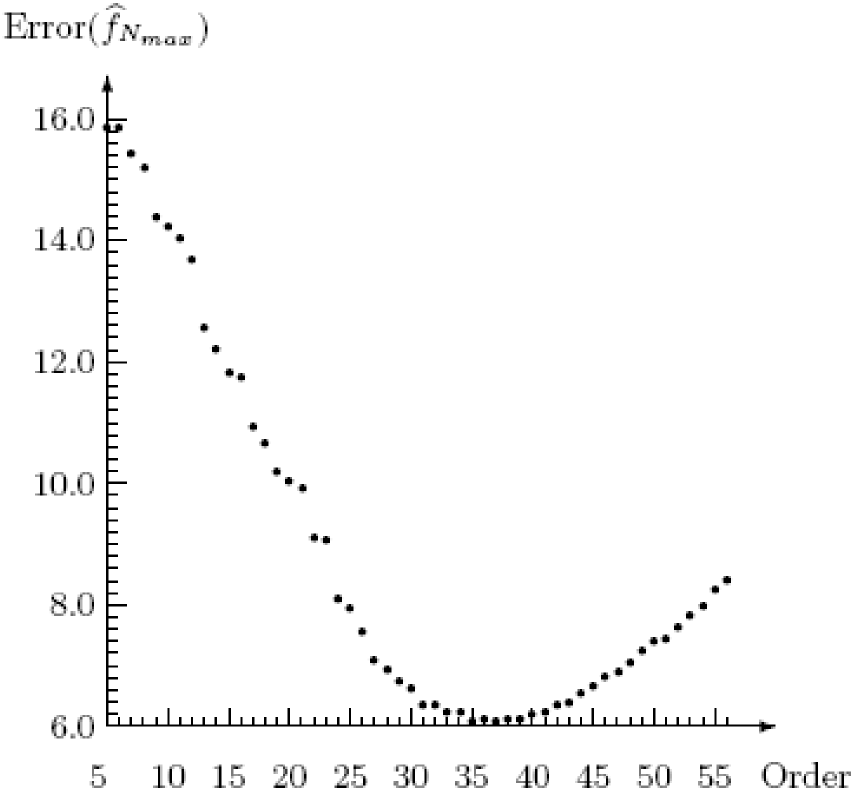
Reconstruction of a noise-free image



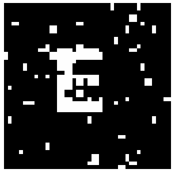


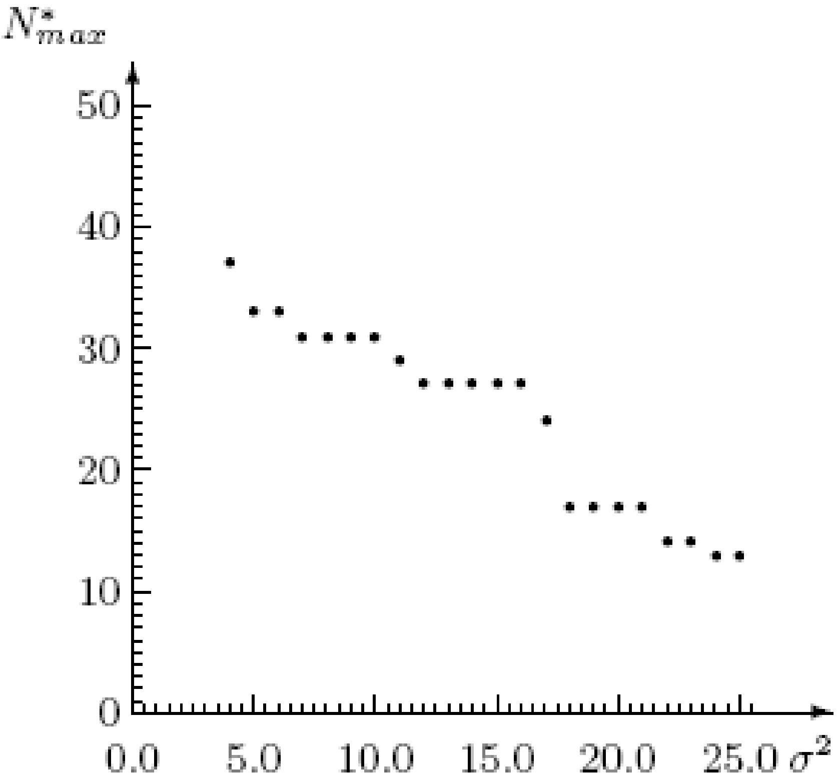
Reconstruction of a noisy image





Reconstruction of a noisy image



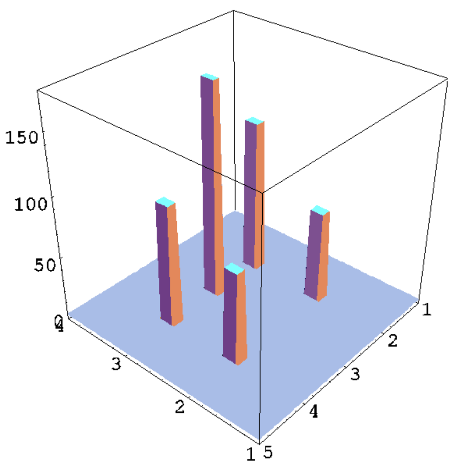


Summary of OG moments

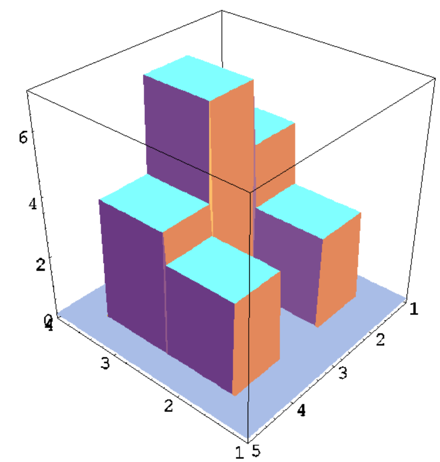
- OG moments are used because of their favorable numerical properties, not because of theoretical contribution
- OG moments should be never used outside the area of orthogonality
- OG moments should be always calculated by recurrent relations, not by expanding into powers
- Moments OG on a square do not provide easy rotation invariance
- Even if the reconstruction from OG moments is seemingly simple, moments are not a good tool for image compression

Algorithms for moment computation

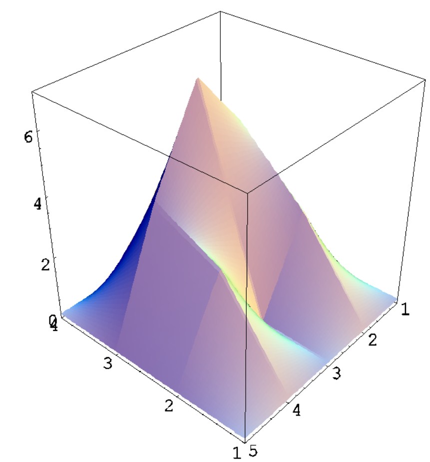
Various definitions of moments in a discrete domain depending on the image model



Sum of Dirac δ-functions

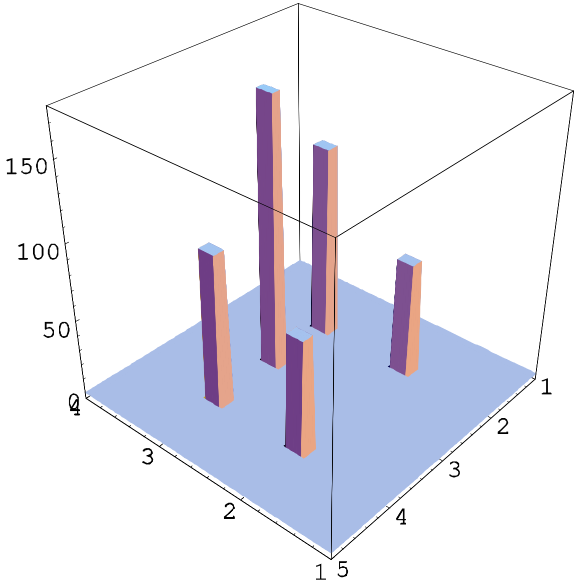


Nearest neighbor interpolation



Bilinear interpolation

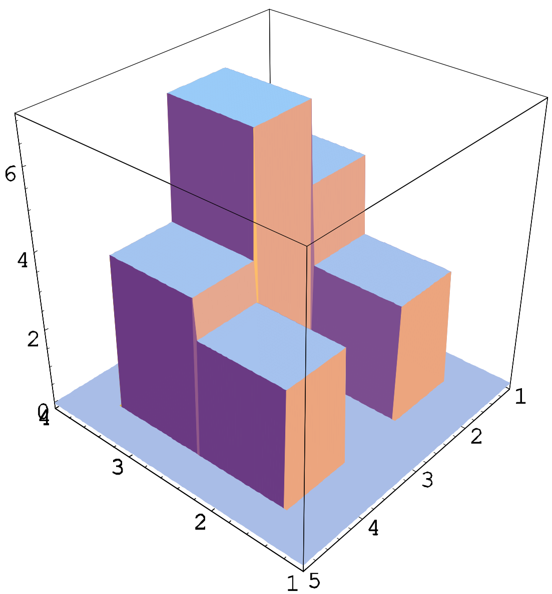
Moments in a discrete domain



$$\bar{m}_{pq} = \sum_{i=1}^{N} \sum_{j=1}^{M} i^p j^q f_{ij}$$

exact formula

Moments in a discrete domain



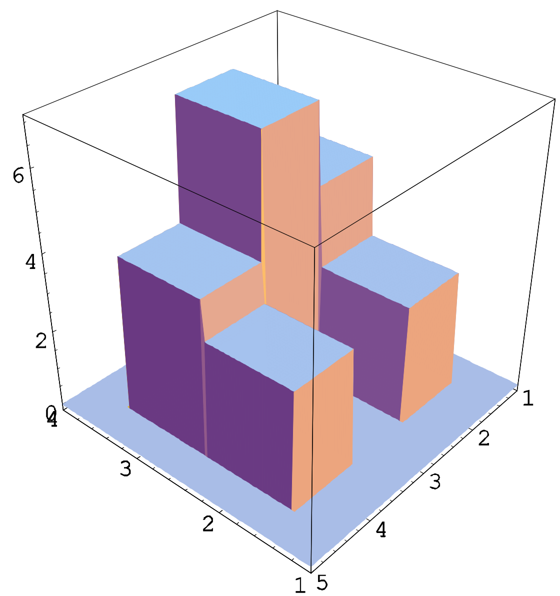
$$\bar{m}_{pq} = \sum_{i=1}^{N} \sum_{j=1}^{M} i^p j^q f_{ij}$$

zero-order approximation

Moments in a discrete domain

$$\hat{m}_{pq} = \sum_{i=1}^{N}\sum_{j=1}^{M} f_{ij} \iint\limits_{S_{ij}} x^p y^q \, \mathrm{d}x \, \mathrm{d}y =$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} f_{ij} \cdot \frac{(i+0.5)^{p+1} - (i-0.5)^{p+1}}{p+1} \cdot \frac{(j+0.5)^{q+1} - (j-0.5)^{q+1}}{q+1}$$



exact formula

Algorithms for binary images

- Decomposition methods
- Boundary-based methods

Decomposition methods

The object is decomposed into *K* disjoint (usually rectangular) "blocks" such that

$$m_{pq}^{(\Omega)} = \sum_{k=1}^{K} m_{pq}^{(B_k)} \qquad K \ll N^2$$

$$m_{pq}^{(B_k)} \sim O(1),$$

$$m_{pq}^{(\Omega)} \sim O(K)$$

Decomposition methods

differ from each other by

- the decomposition algorithms
- the shape of the blocks
- the way how the moments of the blocks are calculated

Delta method (Zakaria et al.)

Decomposition into rows

$$m_{pq}^{G_k} = y_0^q \sum_{i=x_0}^{x_0+\delta-1} i^p$$

Recursive formulae for the summations

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$

$$\sum_{i=1}^{n} i^4 = \frac{n(n+1)(2n+1)(3n^2+3n+1)}{30}$$

$$\sum_{t=1}^{m} \binom{m+1}{t} S_n^t = (n+1)((n+1)^m - 1)$$

where $$S_n^t = \sum_{i=1}^{n} i^t$$

Delta method (Zakaria et al.)

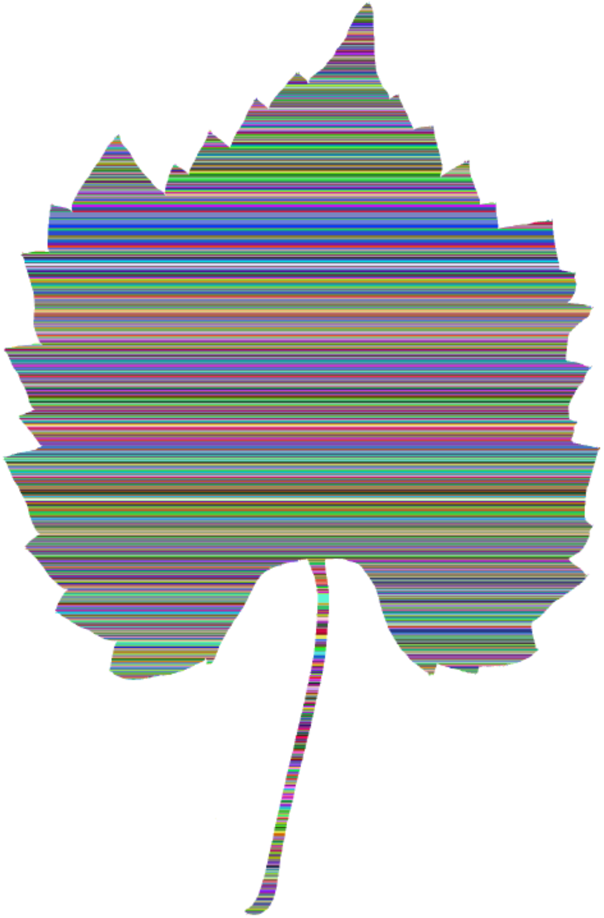
Decomposition into rows

$$m_{pq}^{G_k} = y_0^q \sum_{i=x_0}^{x_0+\delta-1} i^p$$

Simplification by direct integration

$$m_{pq}^{G_k} = y_0^q \int_{x_0}^{x_0+\delta} i^p dx \;\; = \frac{y_0^q}{p+1}[(x_0+\delta)^{p+1} - x_0^{p+1}]$$

Delta method (Zakaria et al.)



Rectangular blocks (Spiliotis et al.)

Decomposition into sets of rows of the same beginning and end

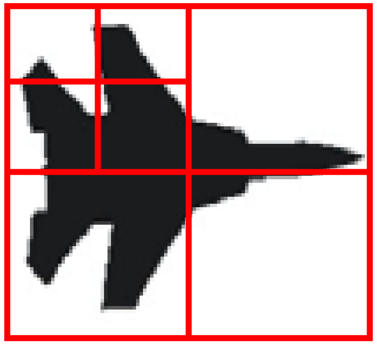
$$m_{pq}^{G_k} = \sum_{i=x_0}^{x_1} \sum_{j=y_0}^{y_1} i^p j^q = \sum_{i=x_0}^{x_1} i^p \cdot \sum_{j=y_0}^{y_1} j^q$$

Simplification by direct integration

$$\frac{1}{(p+1)(q+1)}(x_1^{p+1} - x_0^{p+1})(y_1^{q+1} - y_0^{q+1})$$

Hierarchical decomposition

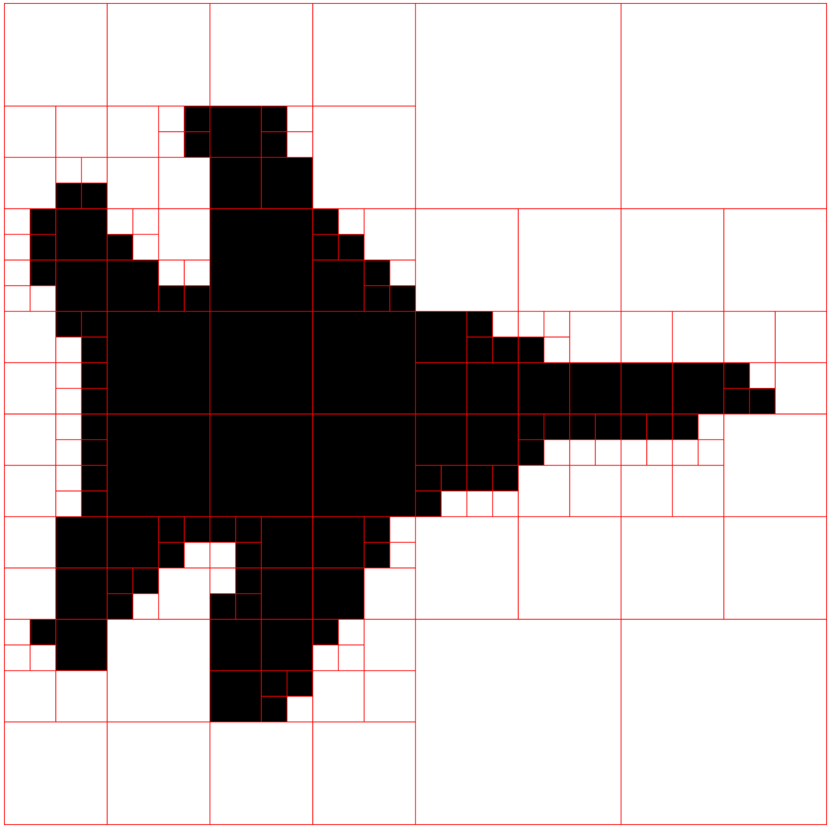
Bin-tree/quad-tree decomposition into homogeneous squares



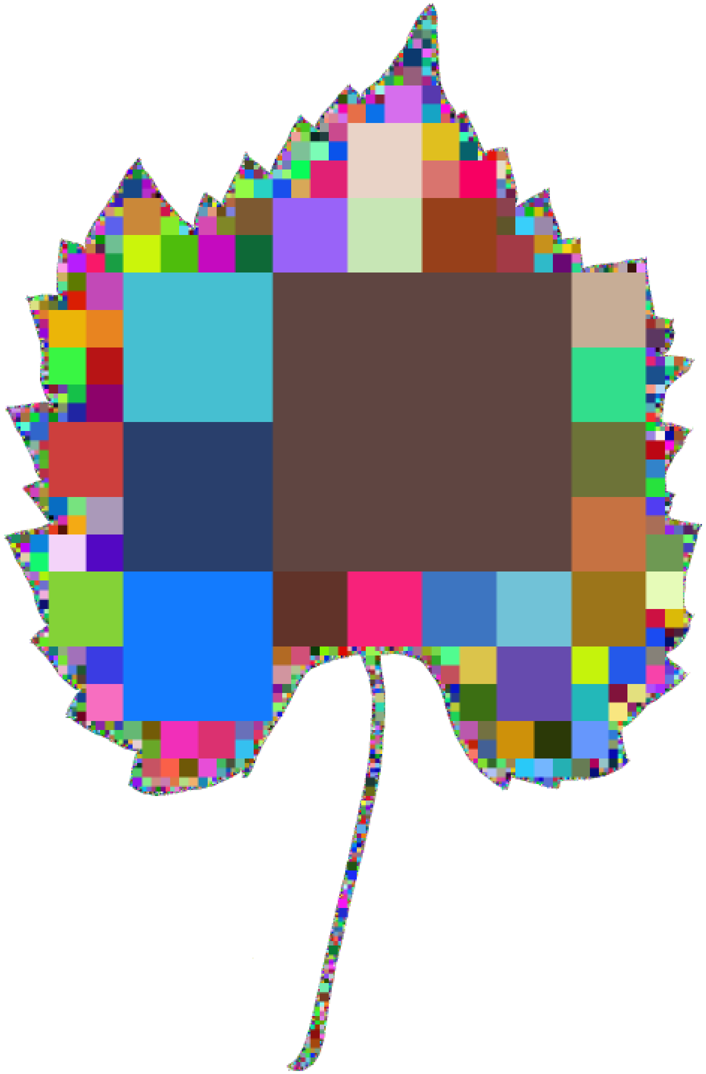
Moment of a block by direct integration

$$\frac{1}{(p+1)(q+1)}(x_1^{p+1} - x_0^{p+1})(y_1^{q+1} - y_0^{q+1})$$

Quadtree decomposition – an example

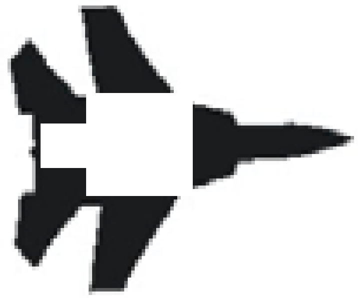


Quadtree decomposition – an example



Morphological decomposition (Sossa et al.)

Recursive decomposition into the "largest inscribed squares"

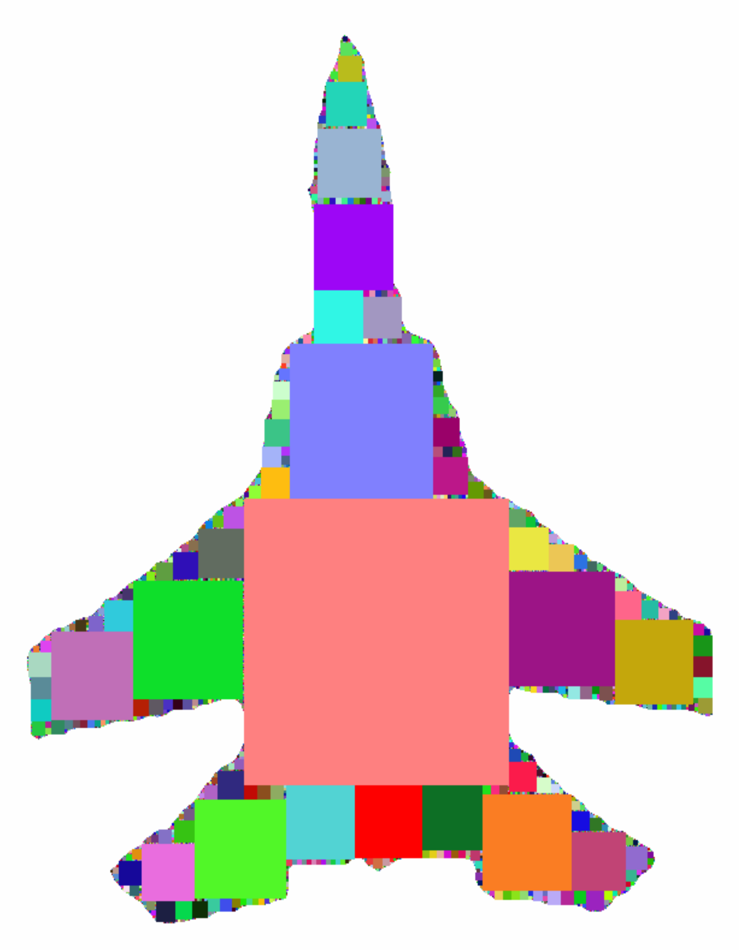


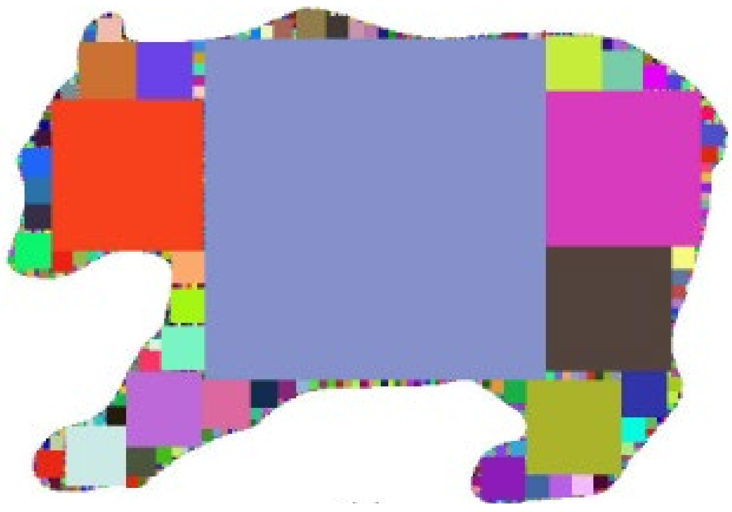
Square centers are found by erosion

Moment of a block by direct integration

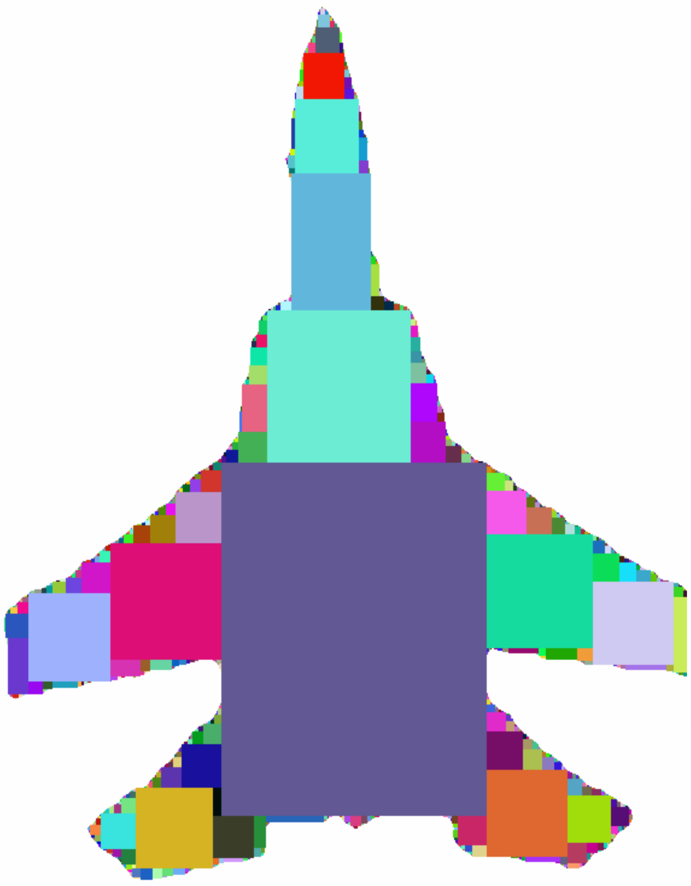
$$\frac{1}{(p+1)(q+1)}(x_1^{p+1} - x_0^{p+1})(y_1^{q+1} - y_0^{q+1})$$

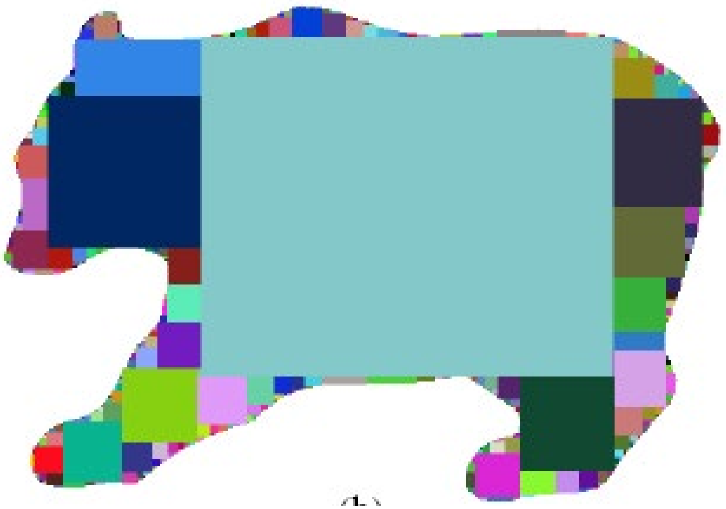
Morphological decomposition into squares





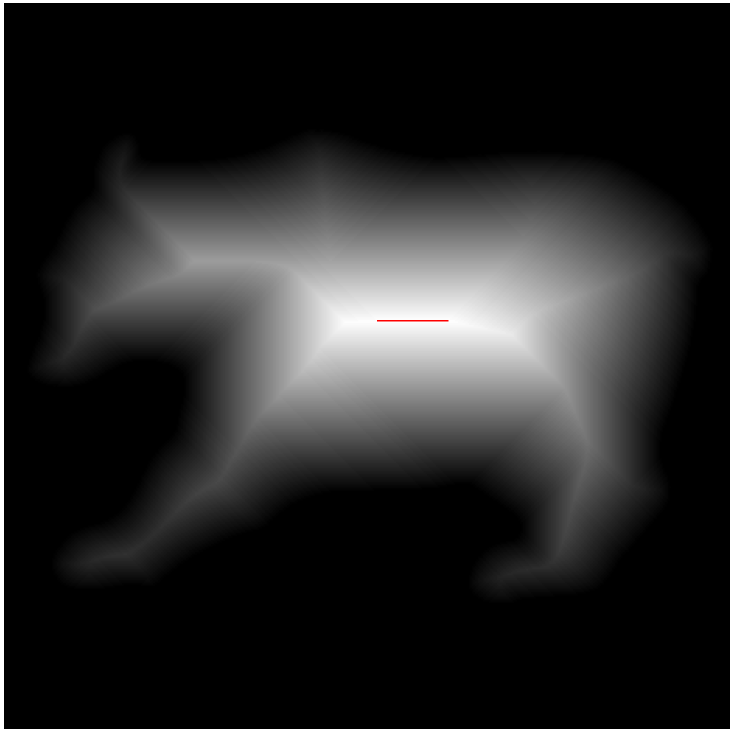
Morphological decomposition into rectangles



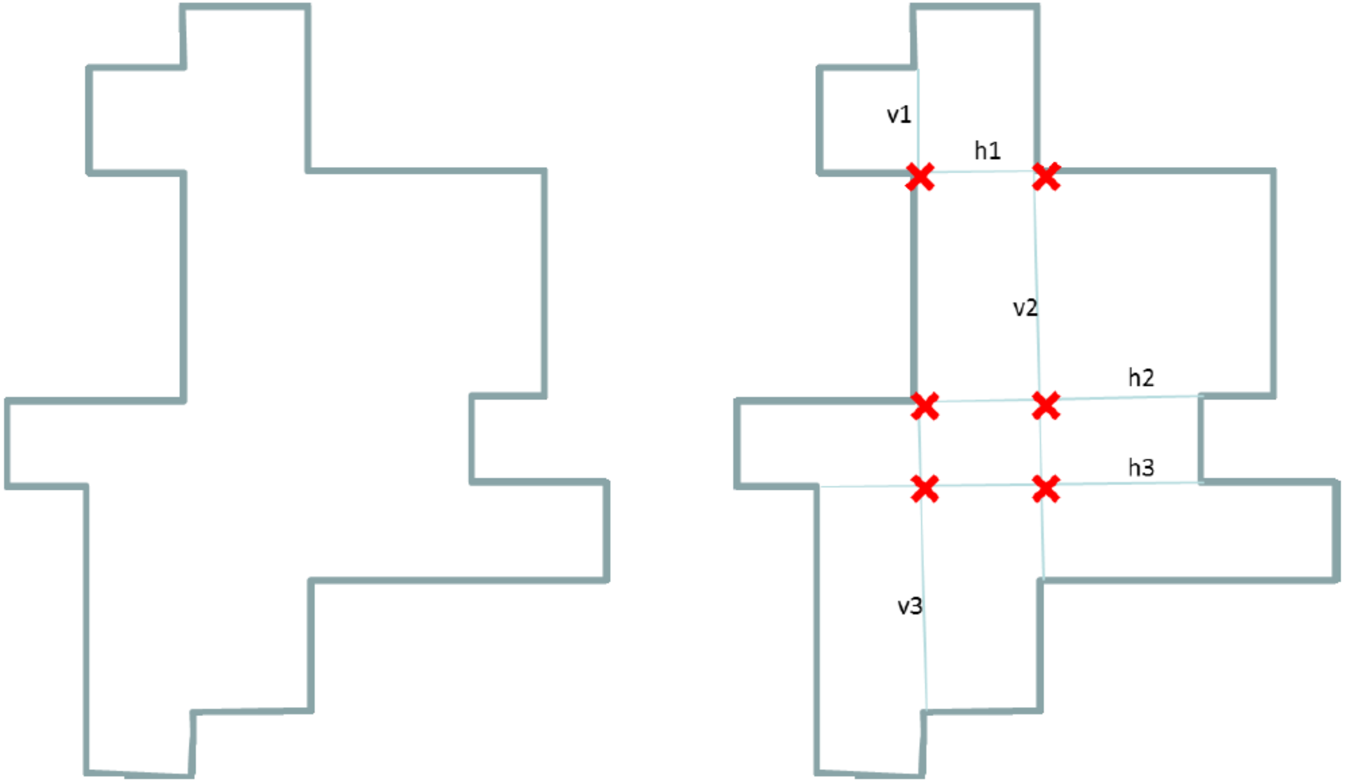


Decomposition by distance transform

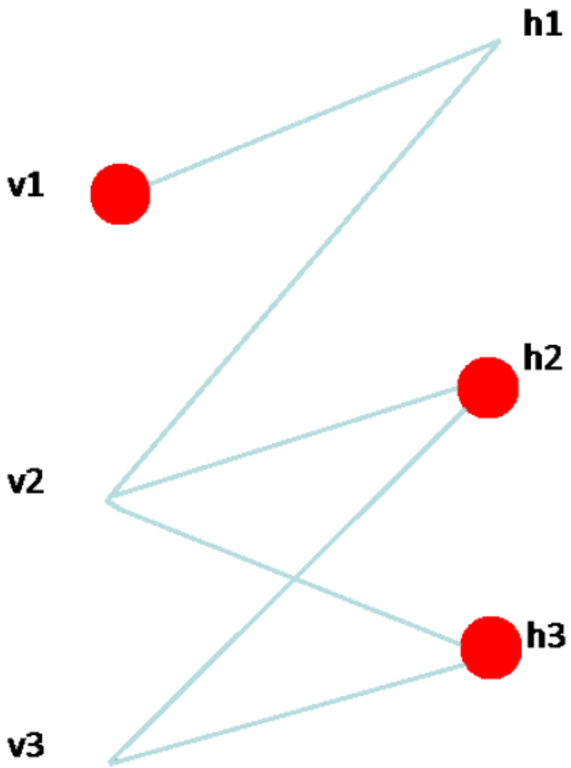
$$d(a,b) = \max\{|a_x - b_x|, |a_y - b_y|\}$$

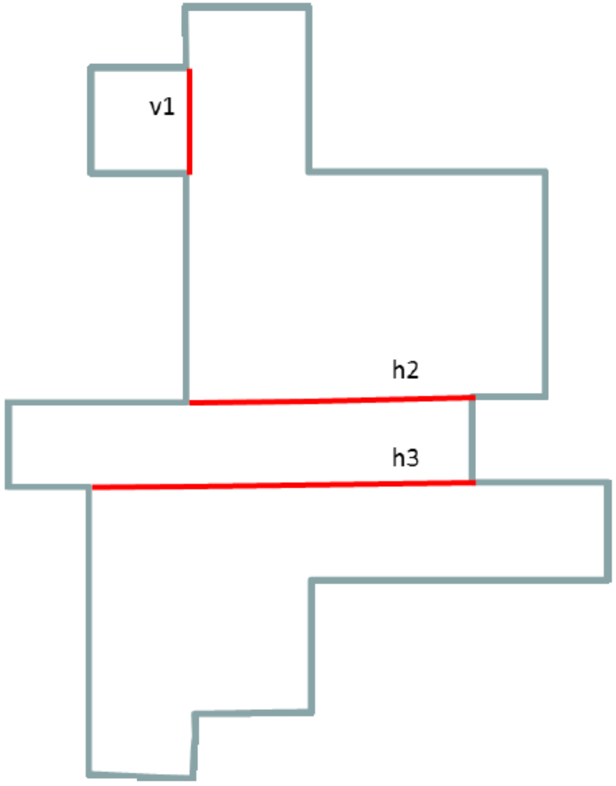


Optimal decomposition

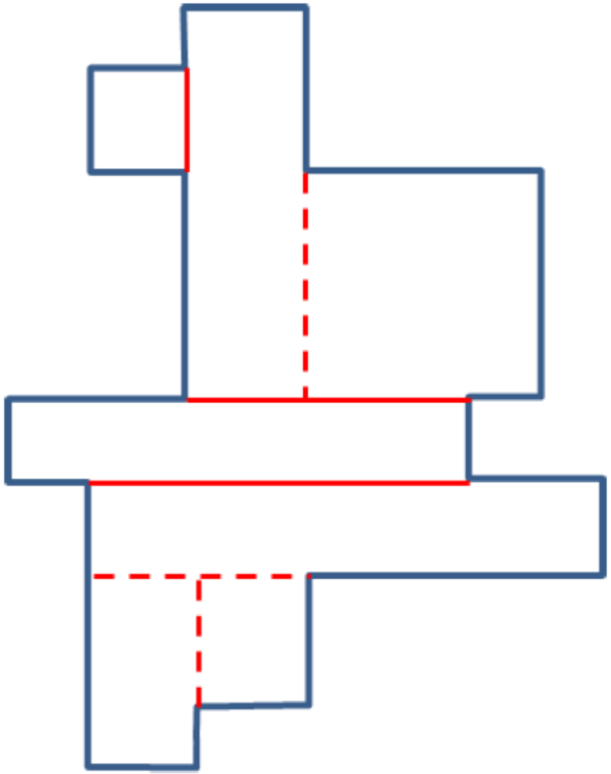


Optimal decomposition

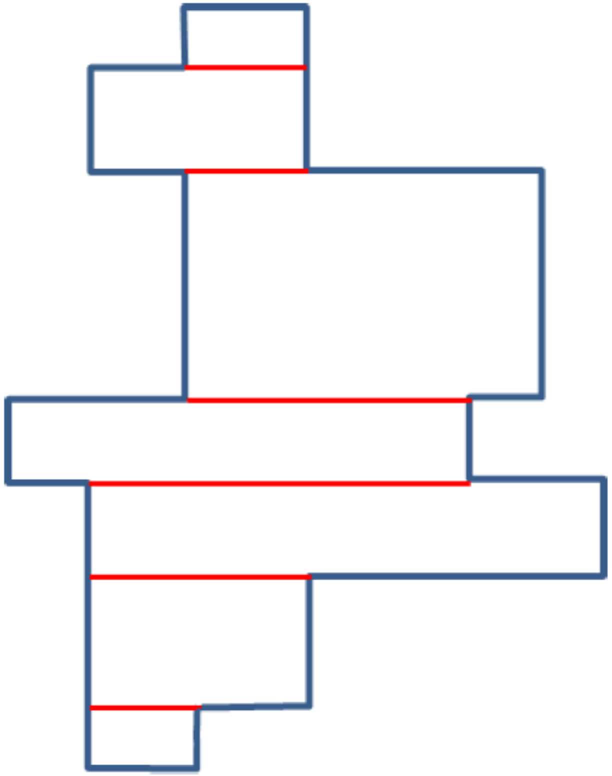




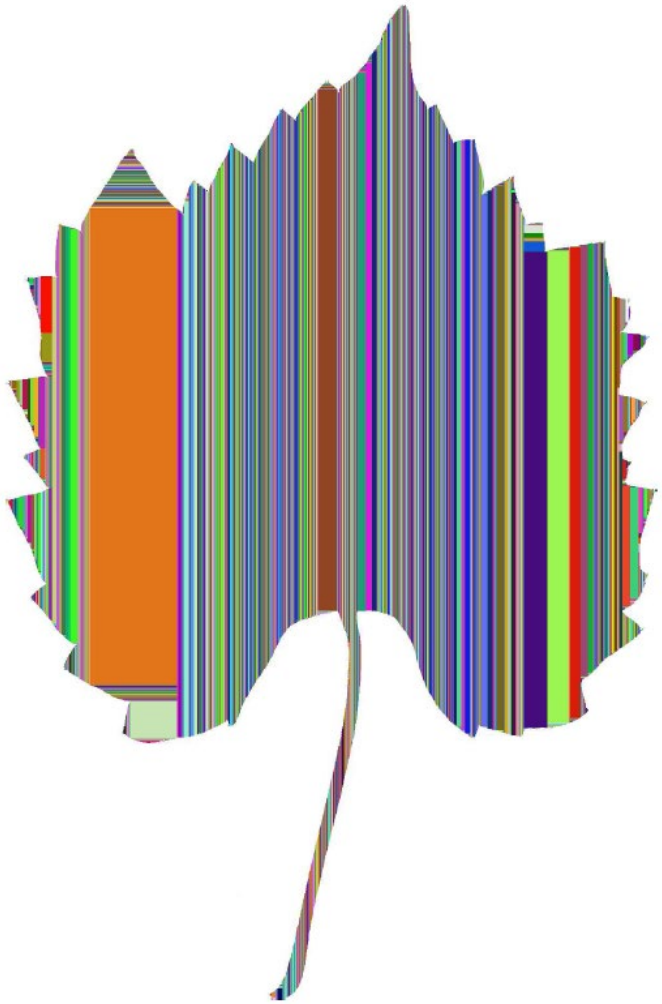
Optimal decomposition



Optimal decomposition



Optimal decomposition



Decompositions – a comparison

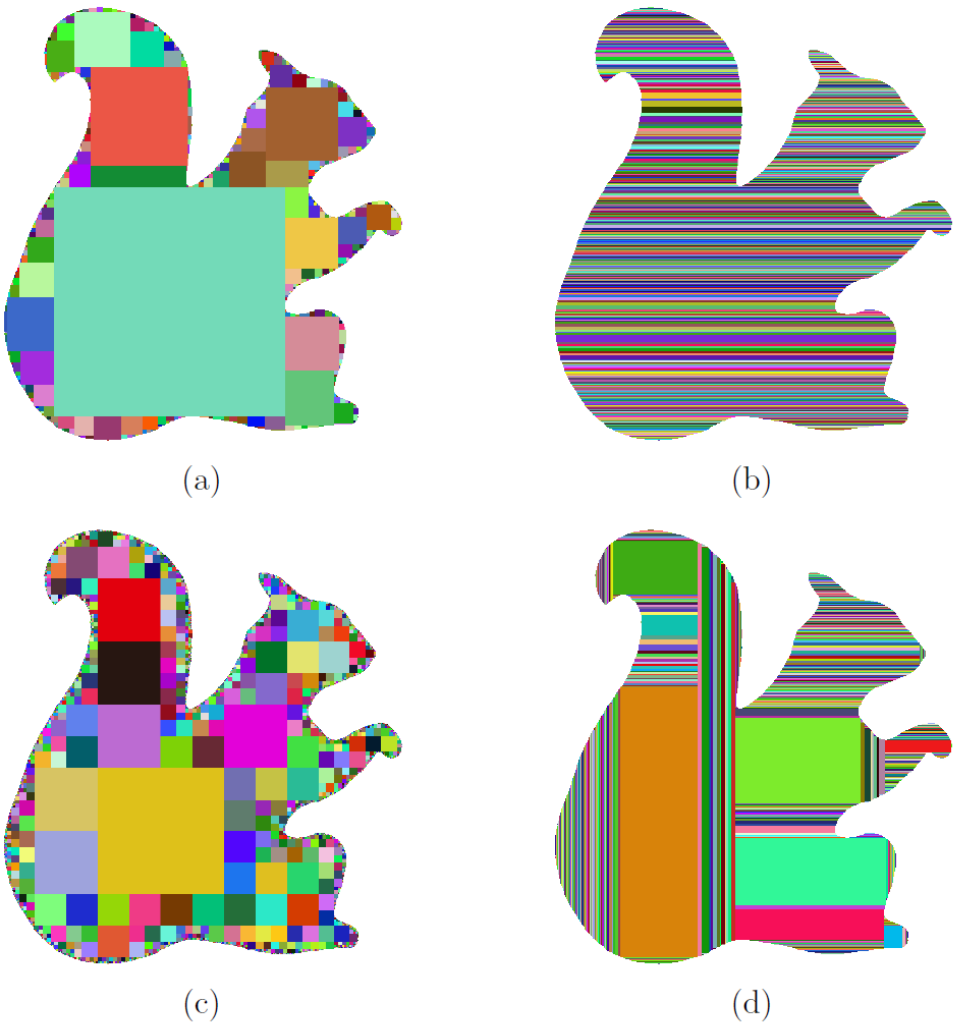
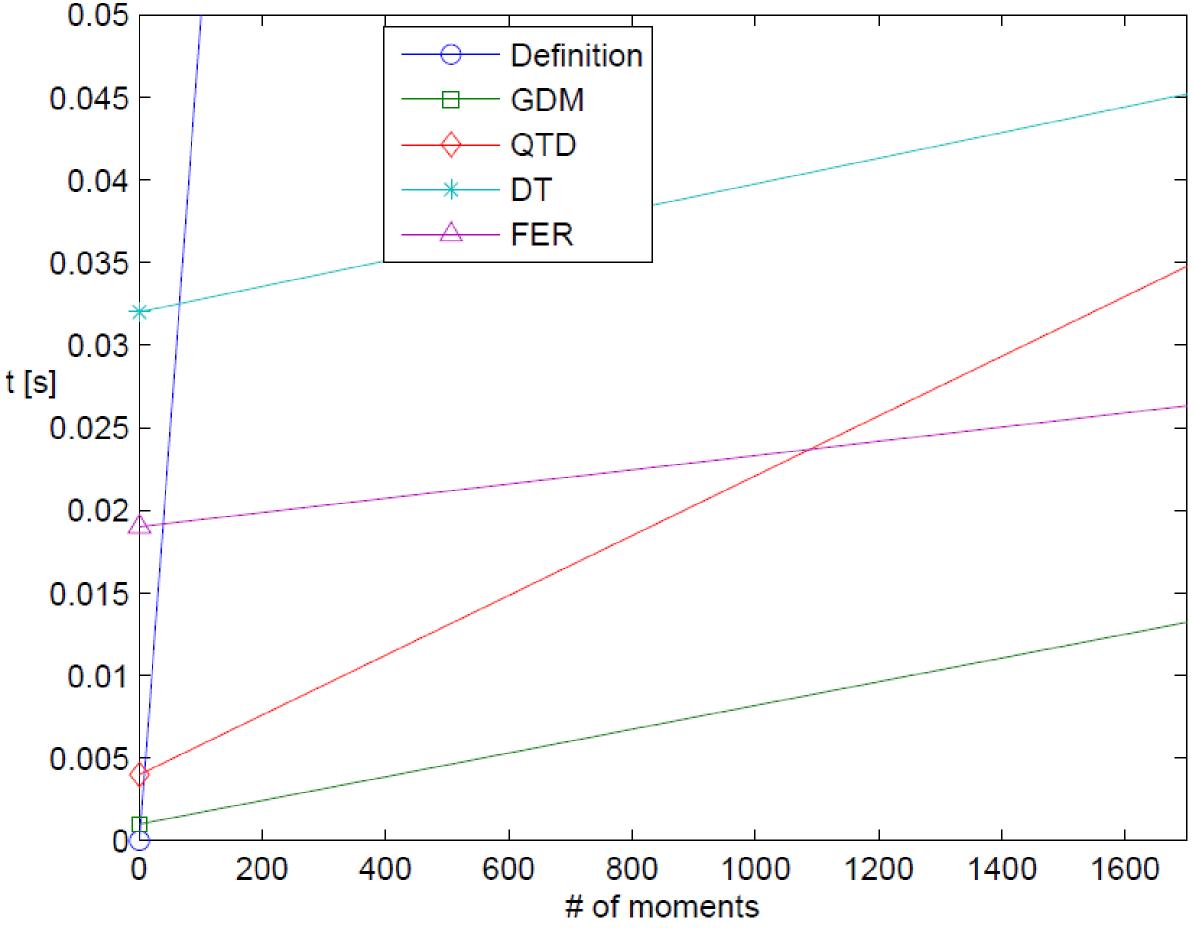
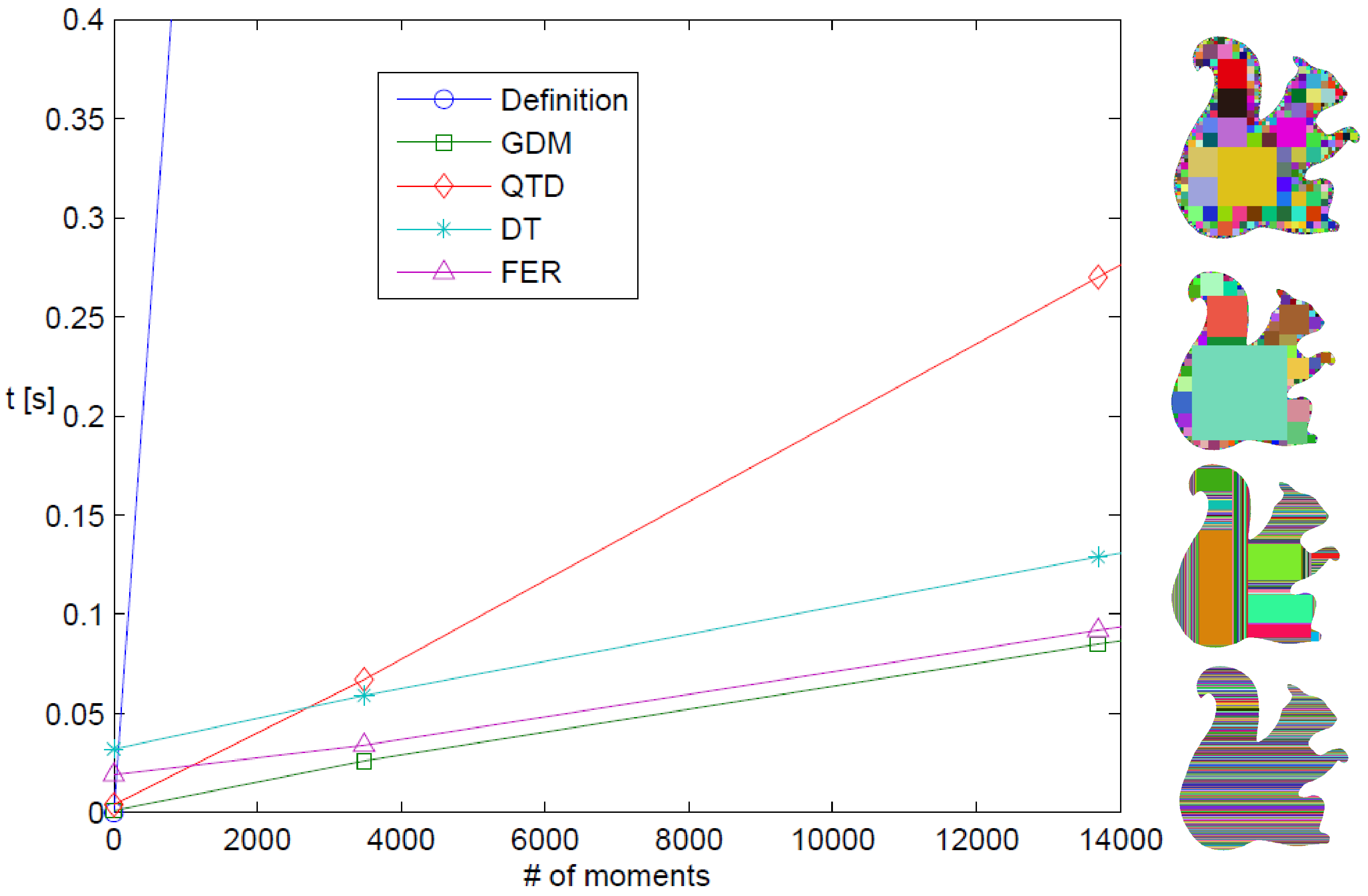


Figure 18: Decomposition of the squirrel image: (a) DT – 697 blocks, (b) GDM – 497 blocks, (c) QTD – 2513 blocks and (d) FER – 476 blocks.

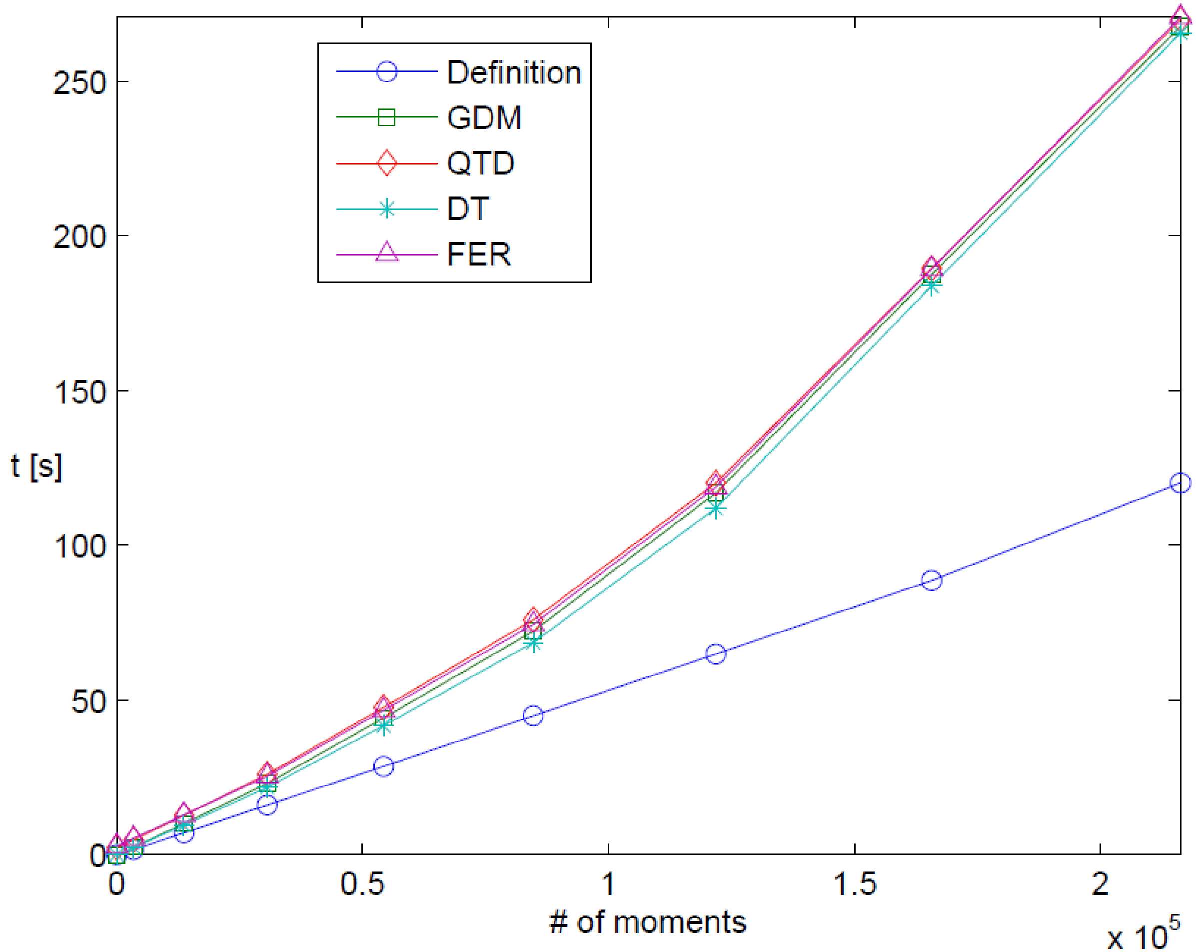
Moment calculation – the squirrel



Moment calculation – the squirrel



Moment calculation – the chessboard



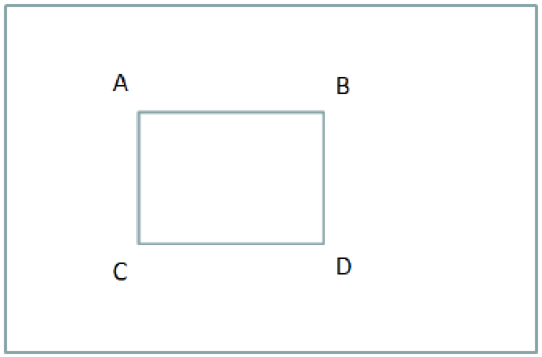
Decomposition methods - complexity

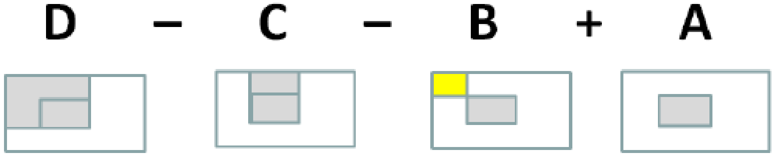
$$m_{pq}^{G} \sim O(K)$$

- Complexity of the decomposition is often ignored (believed to be *O(1)*) but it might be very high – it must be always considered
- Efficient when calculating a large number of moments of the object
- Certain objects cannot be efficiently decomposed at all (a chessboard)

Decomposition for computing convolution

- Convolution with a constant rectangle – *O(1)* per image pixel (with pre-calculations)





Decomposition for computing convolution

- Convolution with a constant rectangle – *O(1)* per image pixel (with pre-calculations)
- Convolution with a binary kernel
  - kernel decomposition into K blocks
  - *O(K)* per pixel

Decomposition for computing convolution

- Convolution with a constant rectangle – *O(1)* per image pixel (with pre-calculations)
- Convolution with a binary kernel
  - kernel decomposition into K blocks
  - *O(K)* per pixel

Mask size	Definition	FFT	Decomposition
35×38	26	4.3	0.96
141×152	411	4.3	0.96

Boundary-based methods

Green's theorem

$$\oint_{\partial\Omega} h(x,y)\ \mathrm{d}x + g(x,y)\ \mathrm{d}y = \iint_{\Omega} \left(\frac{\partial g}{\partial x} - \frac{\partial h}{\partial y} \right)\ \mathrm{d}x\ \mathrm{d}y$$

$$g(x,y) = \frac{x^{p+1} y^q}{p+1},\ \ h(x,y) = 0 \quad \longrightarrow \quad m_{pq}^{(\Omega)} = \frac{1}{p+1} \oint_{\partial\Omega} x^{p+1} y^q\ \mathrm{d}y$$

Calculation of the boundary integral

- Summation pixel-by-pixel
- Polygonal approximation
- Other approximations (splines, etc.)

Discrete Green's theorem (Philips)

$$\bar{m}_{pq}^{(\Omega)} = \sum_{(x,y)\in\partial\Omega+} y^q \sum_{i=1}^{x} i^p - \sum_{(x,y)\in\partial\Omega-} y^q \sum_{i=1}^{x} i^p$$

$$\partial\Omega- = \{(x,y) | (x,y) \notin \Omega, (x+1,y) \in \Omega\}$$

$$\partial\Omega+ = \{(x,y) | (x,y) \in \Omega, (x+1,y) \notin \Omega\}$$

- Equivalent to the delta-method
- Can be simplified by direct integration and further by pre-calculations (efficient for large number of objects)

Boundary-based methods - complexity

- Complexity depends on the length of the boundary
- Detecting boundary is assumed to be fast
- Efficient for objects with simple boundary
- Unlike decomposition methods, they can be used even for small number of moments
- Inefficient for objects with complex boundaries (a chessboard)

Moments of gray-level images

• Decomposition into several binary images (intensity slices, bit planes)

• Approximation of graylevels

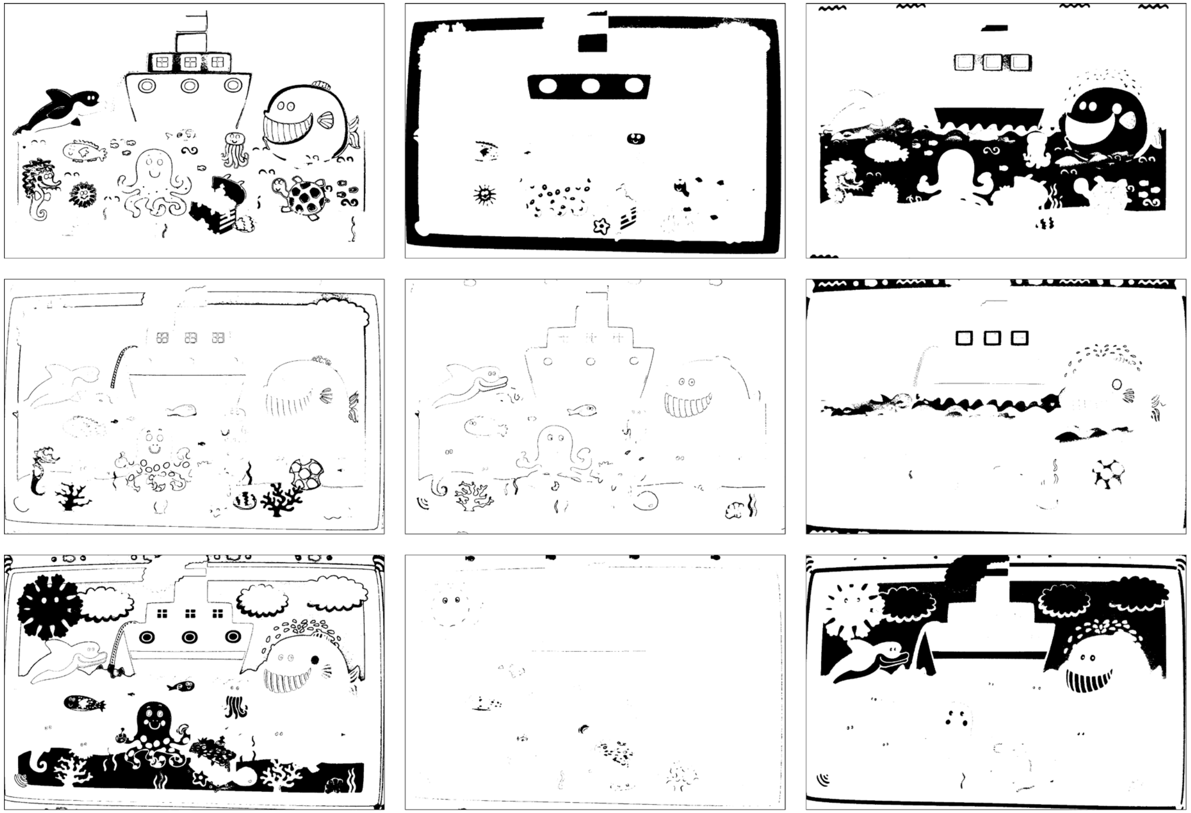
Intensity slicing

$$f(x,y) = \sum_{k=1}^{L-1} k\, f_k(x,y)$$

$$f_k(x,y) = \begin{cases} 1 & \text{if } f(x,y) = k \\ 0 & \text{if } f(x,y) \neq k \end{cases}$$

$$m_{pq}^{(f)} = \sum_{k=1}^{L-1} k\, m_{pq}^{(f_k)}$$

Intensity slicing



Bit-plane slices

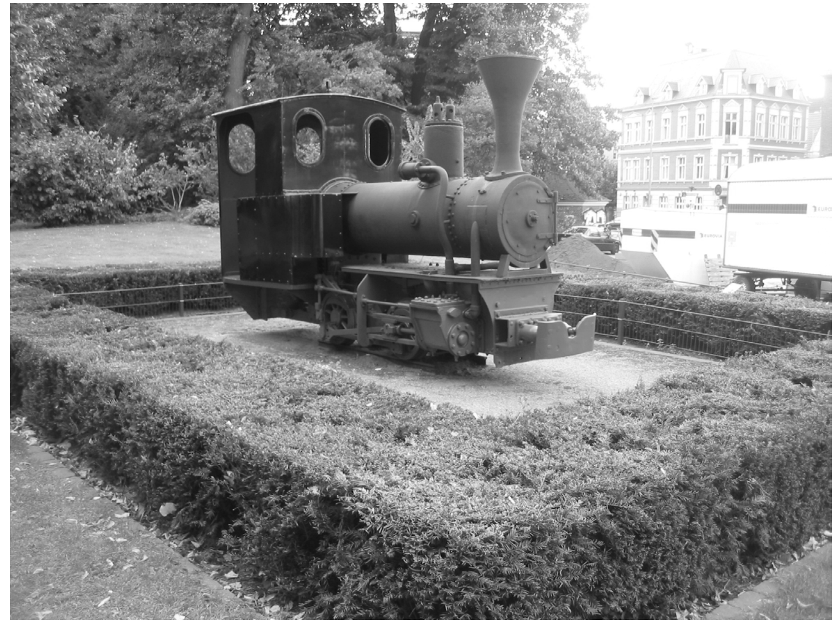
$$f(x,y) = \sum_{k=0}^{\log_2 L - 1} 2^k \, f_k(x,y)$$

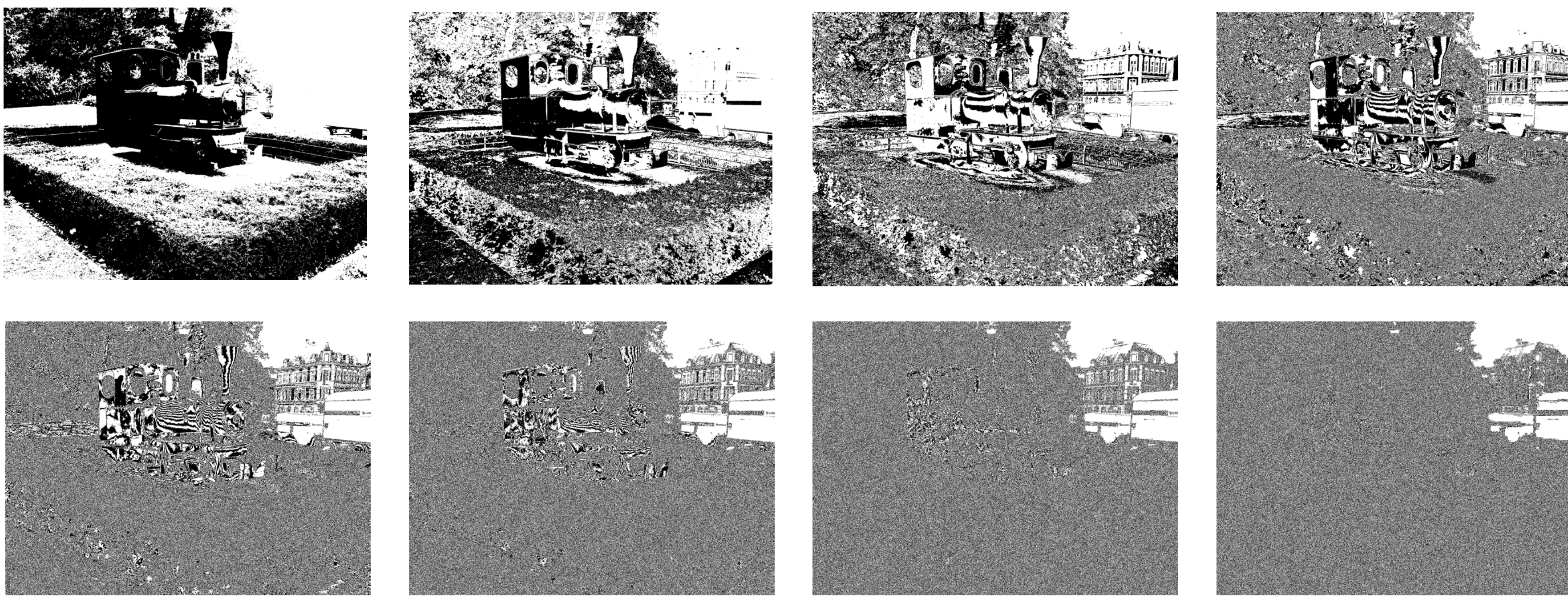
$f_k(x,y)$ is the k-th bit plane of the image

$$m_{pq}^{(f)} = \sum_{k=0}^{\log_2 L - 1} 2^k \, m_{pq}^{(f_k)}$$

Low bit planes are often ignored

Bit-plane slices





A detail of the zero-bit plane

Approximation methods

The image is decomposed into blocks where it can be approximated by an "easy-to-integrate" function (e.g. by polynomials)

Any kind of decomposition can be used.

Polynomial approximation of graylevels

$$P_n(x,y) = \sum_{\substack{k=0 \\ \ell+k<n}}^{n} \sum_{\ell=0}^{n} a_{k\ell} x^k y^\ell$$

$$\begin{aligned} \hat{m}_{pq}^{(B)} &= \int_0^M \int_0^N x^p y^q P_n(x,y) \, \mathrm{d}x \, \mathrm{d}y \\ &= \sum_{\substack{k=0 \\ \ell+k\leq n}}^{n} \sum_{\ell=0}^{n} a_{k\ell} \frac{M^{p+k+1} N^{q+\ell+1}}{(p+k+1)(q+\ell+1)} \end{aligned}$$

Algorithms for OG moments

Specific methods

- Methods using recurrent relations
- Decomposition methods
- Boundary-based methods

Are moments good features?

- YES

 - well-developed mathematics behind, invariance to many transformations

 - complete and independent set

 - good discrimination power

 - robust to noise

- NO

 - moments are global

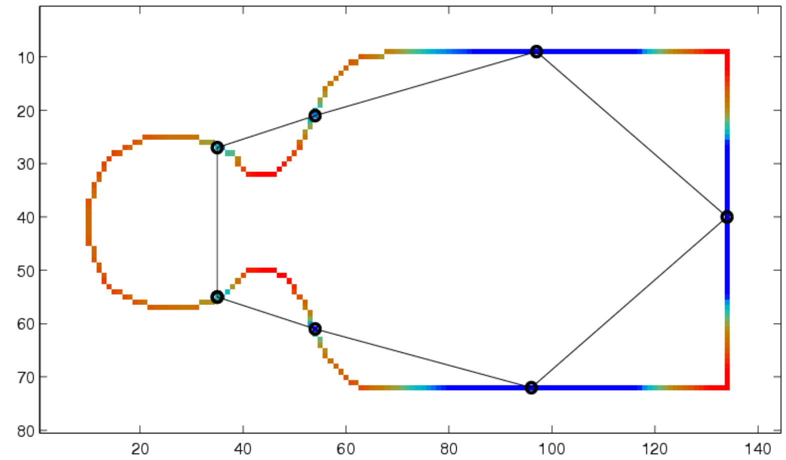
 - small local disturbance affects all moments

 - careful object segmentation is required

How to make the moment invariants local?

Dividing the object into invariant parts

- Inflection points and centers of straight lines are affine invariants



- Computing the AMI's of each part
- Recognition via maximum substring matching

Thank you !

Any questions?