Front-End Vision: A Multiscale Geometry Engine

Bart M. ter Haar Romeny[1], and Luc M.J. Florack[2]

Utrecht University, the Netherlands
[1] Image Sciences Institute, PO Box 85500, 3508 TA Utrecht
[2] Department of Mathematics, PO Box 80010, 3508 TA Utrecht
B.terHaarRomeny@isi.uu.nl, Luc.Florack@math.uu.nl

Abstract. The paper is a short tutorial on the multiscale differential geometric possibilities of the front-end visual receptive fields, modeled by Gaussian derivative kernels. The paper is written in, and interactive through the use of Mathematica 4, so each statement can be run and modified by the reader on images of choice. The notion of multiscale invariant feature detection is presented in detail, with examples of second, third and fourth order of differentiation.

1 Introduction

The front end visual system belongs to the best studied brain areas. Scale-space theory, as pioneered by Iijima in Japan [10,17] and Koenderink [11] has been heavily inspired by the important derivation of the Gaussian kernel and its derivatives as regularized differential operators, and the linear diffusion equation as its generating partial differential equation. To view the front-end visual system as a 'geometry-engine' is the inspiration of the current work. Simultaneously, the presented examples of applications of (differential) geometric operations may inspire operational models of the visual system.

Scale-space theory has developed into a serious field [8, 13]. Several comprehensive overview texts have been published in the field [5, 7, 12, 15, 18]. So far, however, this robust mathematical framework has seen impact on the computer vision community, but there is still a gap between the more physiologically, psychologically and psychophysically oriented researchers in the vision community. One reason may be the nontrivial mathematics involved, such as group invariance, differential geometry and tensor analysis.

The last couple of years symbolic computer algebra packages, such as Mathematica, Maple and Matlab, have developed into a very user friendly and high level prototyping environment. Especially Mathematica combines the advantages of symbolic manipulation and processing with an advanced front-end text processor. This paper highlights the use of Mathematica 4.0 as an interactive tutorial toolkit for easy experimenting and exploring front-end vision simulations. The exact code can be used rather then pseudocode. With these high level programming tools most programs can be expressed in very few lines, so it keeps the reader at a highly intuitive but practical level. Mathematica notebooks are portable, and run on *any* system equivalently. Previous speed limitations are now well overcome. The full (1400 pages) documentation is available online, (see www.wri.com).

1.1 Biological Inspiration: Receptive Field Profiles from First Principles

It is well known that the Gaussian kernel, $G(\vec{x},\sigma)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{\vec{x}.\vec{x}}{2\sigma^2})$ as a front-end visual measurement aperture can be uniquely derived in quite a number of ways (for a comprehensive overview see [17]). These include the starting point that lower resolution levels have a higher resolution level as cause ('causality' [11]), or that there is linearity and no preference for location, orientation and size of the aperture ('first principles' [2]). This Gaussian kernel is the Green's function of the linear, isotropic *diffusion equation* $\frac{\partial^2 L}{\partial x^2}+\frac{\partial^2 L}{\partial y^2}=L_{xx}+L_{yy}=\frac{\partial L}{\partial s}$, where $s=2\sigma^2$ is the variance. Note that the derivative to scale is here the derivative to σ^2 which also immediately follows from a consideration of the dimensionality of the equation. All *partial derivatives* of the Gaussian kernel are solutions too of the diffusion equation.

The Gaussian kernel and all of its partial derivatives form a one-parameter *family* of kernels where the scale σ is the free parameter. This is a general feature of the biological visual system: the exploitation of *ensembles* of aperture functions, which are mathematically modeled by families of kernels for a free parameter, e.g. for all scales, derivative order, orientation, stereo disparity, motion velocity etc.

The Gaussian kernel is the unique kernel that generates no *spurious resolution* (e.g. the squares so familiar with zooming in on pixels). It is the physical *point operator*, the Gaussian derivatives are the physical *derivative operators*, at the scale given by the Gaussian standard deviation.

The receptive fields in the primary visual cortex closely resemble Gaussian derivatives, as was first noticed by Young [19] and Koenderink [11]. These RF's come at a wide range of sizes, and at all orientations.

Below two examples are given of measured receptive field sensitivity profiles of a cortical *simple cell* (left) and a Lateral Geniculate Nucleus (LGN) *center-surround* cell, measured by DeAngelis, Ohzawa and Freeman [1], http://totoro.berkeley.edu/.

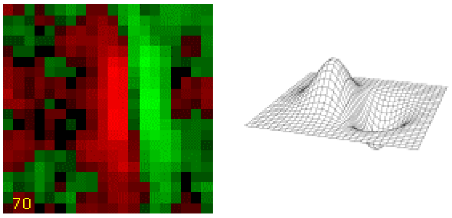
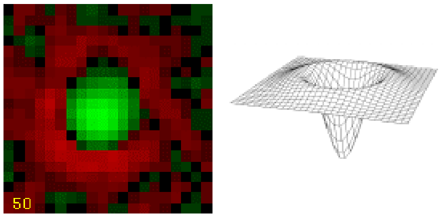


Fig. 1. a. Cortical simple cell, modeled by a first order Gaussian derivative. From[1].



b. Center-surround LGN cell, modeled by Laplacean of Gaussian. From [1].

Through the center-surround structure at the very first level of measurement on the retina the *Laplacean* of the input image can be seen to be taken. The linear diffusion equation states that this Laplacean is equal to the first derivative to scale: $\frac{\partial^2 L}{\partial x^2}+\frac{\partial^2 L}{\partial y^2}=\frac{\partial L}{\partial s}$. One conjecture for its presence at this first level of observation might be that the visual system actually measures L_s, i.e. the change in signal ∂L when the aperture is changed with ∂s: at homogeneous areas there is no output, at highly textured areas there is much output. Integrating both sides of $\partial L = (L_{xx}+L_{yy})\, \partial s$ over all scales gives the measured intensity in a robust fashion.

The derivative of the observed (convolved) data $\frac{\partial}{\partial x}(L \otimes G) = L \otimes \frac{\partial G}{\partial x}$ shows that differentiation and observation is accomplished in a single step: convolution with a Gaussian derivative kernel. Differentiation is now done by *integration*, i.e. by the convolution integral.

The Gaussian kernel is the physical analogon of a mathematical point, the Gaussian derivative kernels are the physical analogons of the mathematical differential operators. Equivalence is reached for the limit when the scale of the Gaussian goes to zero: $\lim_{\sigma \to 0} G(x,\sigma) = \delta(x)$, where is the Dirac delta function, and $\lim_{\sigma \to 0} \otimes G(x,\sigma) = \frac{\partial}{\partial x}$. Any differention blurs the data somewhat, with the amount of the scale of the differential operator. There is no way out this increase of the inner scale, we can only try to minimize the effect. The Gaussian kernel has by definition a strong *regularizing* effect [16,12].

2 Multiscale Derivatives

It is essential to work with descriptions that are *independent of the choice of coordinates*. Entities that do not change under a group of coordinate transformations are called *invariants* under that particular group. The only geometrical entities that make physically sense are invariants. In the words of Hermann Weyl: any invariant has a specific geometric *meaning*. In this paper we only study orthogonal and affine invariants. We first build the operators in Mathematica:

The function **gDf[im,nx,ny, σ]** implements the convolution of the image with the Gaussian derivative for 2D data in the Fourier domain. This is an exact function, no approximations other then the finite periodic window in both domains. Variables: `im` = 2D image (as a list structure), `nx,ny` = order of differentiation to *x* resp. *y*, **σ** = scale of the kernel, in pixels.

```
gDf[im_,nx_,ny_,σ_]:= Module[{xres, yres, gdkernel},
 {yres, xres} = Dimensions[im];
 gdkernel = N[Table[Evaluate[
 D[1/(2 Pi σ^2) Exp[-((x^2+y^2)/(2σ^2))],{x,nx},{y, ny}]],
 {y,-(yres-1)/2,(yres-1)/2}, {x,-(xres-1)/2,(xres-1)/2}]];
 Chop[N[Sqrt[xres yres] InverseFourier[Fourier[im]
 Fourier[RotateLeft[gdkernel, {yres/2, xres/2}]]]]]];
```

This function is rather slow, but is exact. A much faster implementation exploits the separability of the Gaussian kernel, and this implementation is used in the sequel:

```
gD[im_,nx_,ny_,σ_] := Module[{x,y,kx,ky,tmp},
   kx ={N[Table[Evaluate[D[(Exp[-(x^2/(2σ^2))])/
(σ Sqrt[2π]),{x,nx}]],{x,-4σ, 4σ}]]};
```

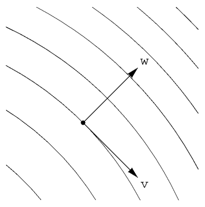
```
   ky = {N[Table[Evaluate[D[(Exp[-(y²/(2σ²))])/
(σ Sqrt[2π]), {y, ny}]], {y, -4σ, 4σ}]]};
tmp = ListConvolve[kx, im, Ceiling[Dimensions[kx]/2]];
Transpose[ListConvolve[ky,
Transpose[tmp],Reverse[Ceiling[Dimensions[ky]/2]]]]];
```

An example is the gradient $\sqrt{L_x^{\ 2} + L_y^{\ 2}}$ at a range of scales:

```
im = Import["mr128.gif"][[1,1]];
p1 = Table[grad = Sqrt[gD[im,1,0,σ]² + gD[im,0,1,σ]²];
{ListDensityPlot[grad,PlotRange→{0, 40}, DisplayFunction→
Identity],
 ListDensityPlot[σ grad, PlotRange→{0, 40},
DisplayFunction→Identity]}, {σ, 1, 5}];
Show[GraphicsArray[Transpose[p1]]];
```

3 Gauge Coordinates

In order to establish differential geometric properties it is easiest to exploit intrinsic geometry. I.e. that we define a new coordinate frame for our geometric explorations which is related to the local isophote structure, so it is different in every different point. A straightforward definition of a new local coordinate frame in 2D is where we cancel the degree of freedom of rotation by defining gauge coordinates: we locally 'fix the gauge'.



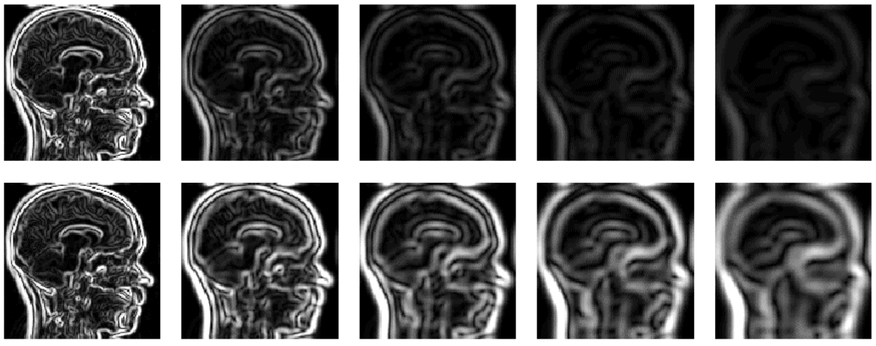


Fig. 2. Top row: Gradient of a sagittal MR image at scales 1, 2, 3, 4 and 5 pixels. Lower row: same scales, gradient in natural dimensionless coordinates, i.e. x'→x/σ, so $\partial / \partial x' \mapsto \sigma \, \partial / \partial x$. This leaves the intensity range of differential features more in a similar output range due to scale invariance. Resolution 128^2.

The 2D unit vector frame of gauge coordinates $\{v,w\}$ is defined as follows: v is the unit vector in the direction tangential to the isophote, so $L_v \equiv 0$, w is defined perpendicular to v, i.e. in the direction of the intensity gradient.

The derivatives to v and w are by definition features that are invariant under orthogonal transformations, i.e. rotation and translation. To apply these gauge derivative operators on images, we have to convert to the Cartesian $\{x,y\}$ domain. The derivatives to v and w are defined as:

$$\partial_v = \frac{-L_y\partial_x + L_x\partial_y}{\sqrt{L_x^{\ 2} + L_y^{\ 2}}} = L_i\varepsilon_{ij}\partial_j \quad ; \quad \partial_w = \frac{L_x\partial_x + L_y\partial_y}{\sqrt{L_x^{\ 2} + L_y^{\ 2}}} = L_i\delta_{ij}\partial_j$$

The second formulation uses *tensor notation*, where the indices i,j stand for the range of dimensions. δ_{ij} is the nabla operator, δ_{ij} and ε_{ij} are the symmetric Kronecker tensor and the antisymmetric Levi-Civita tensor respectively (in 2D). The definitions above are easily accomplished in Mathematica:

```
δ = IdentityMatrix[2]
ε = Table[Signature[{i,j}],{j,2},{i,2}]
jacobean = Sqrt[Lx^2 + Ly^2];
dv = 1/jacobean*{Lx,Ly}.ε.{D[#1,x],D[#1,y]}&
dw = 1/jacobean*{Lx,Ly}.δ.{D[#1,x],D[#1,y]}&
```

The notation `(...#)&` is a 'pure function' on the argument `#`, e.g. `D[#,x]&` takes the derivative. Now we can calculate any derivative to v or w by applying the operator *dw* or *dv* repeatedly. Note that the *Lx* and *Ly* are constant terms.

```
rule1 = {Lx → ∂x L[x,y], Ly → ∂y L[x,y]};
rule2 = Derivative[n_,m_][L][x,y] → L <>
   Table[x,{n}] <> Table[y, {m}];
Lw = dw[L[x, y]] /. rule1 /. rule2 // Simplify
```

$$\sqrt{Lx^2 + Ly^2}$$

```
Lww = dw[dw[L[x, y]]] /. rule1 /. rule2 // Simplify
```

$$\frac{Lx^2 Lxx + 2 Lx Lxy Ly + Ly^2 Lyy}{Lx^2 + Ly^2}$$

Due to the fixing of the gauge by removing the degree of freedom for rotation, we have an important result: *every derivative to v and w is an orthogonal invariant*, i.e. an invariant property where translation or rotation of the coordinate frame is irrelevant. It means that polynomial combinations of these gauge derivative terms are invariant. We now have the toolkit to make gauge derivatives to any order.

3.1 Examples to 2nd, 3rd and 4th Order

The definitions for the gauge differential operators ∂_v and ∂_w need to have their regular differential operators be replaced by Gaussian derivative operators. To just show the textual formula, we do not yet evaluate the derivative by using temporarily HoldForm (`/.` means 'apply rule'):

```
gauge2D[im_, nv_, nw_, σ_] := (Nest[dw,Nest[dv,L[x,y],nv],nw]
/. {Lx -> ∂x L[x,y], Ly -> ∂y L[x,y]})
/. ((Derivative[n_, m_])[L])[x,y] -> HoldForm[gD[im, n, m,
σ]] // Simplify)
```

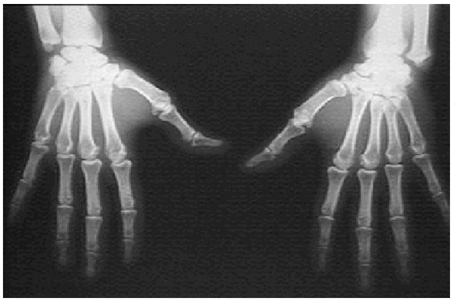
3.2 Ridge Detection

L_{vv} is a good ridge detector. Here is the Cartesian (in {x,y}) expression for L_{vv}:

```
Clear[im,σ]; gauge2D[im, 2, 0, 2]

(gd[im,0,2,2] gd[im,1,0,2]^2 - 2 gd[im,0,1,2] gd[im,1,0,2]
gd[im,1,1,2] + gd[im,0,1,2]^2 gd[im,2,0,2]) /
(gd[im,0,2,2]^2+gd[im,0,2,2]^2)

im = Import["hands.gif"][[1, 1]];
Lvv = gauge2D[im, 2, 0, 3] // ReleaseHold;
Block[{$DisplayFunction = Identity},p1 = ListDensityPlot[im];
p2 = ListDensityPlot[Lvv];]; Show[GraphicsArray[{p1, p2}]];
```



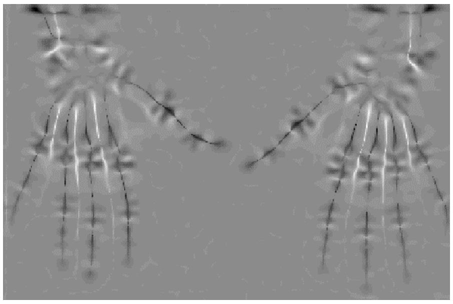


Fig. 3. a. Input image: X-ray of hands, resolution 439x138 pixels. **b.** Ridge detection with L_{vv}, scale 3 pixels. Note the concave and convex ridges.

3.3 Affine Invariant Corner Detection

Corners can be defined as locations with high isophote curvature κ and high intensity gradient L_w. Isophote curvature κ is defined as the change *w”* of the tangent vector *w’* in the gauge coordinate system. When we differentiate the definition of the isophote (*L* = Constant) to *v*, we find $\kappa = -Lvv/Lw$:

```
D[L[v, w[v]] = = Constant, v]; κ=
w''[v]/.Solve[D[L[v,w[v]]==Constant,{v,2}]/.w'[v]→0,w''[v]]

-L^(0,2)[v,w[v]]/L^(1,0)[v,w[v]]
```

Blom proposed as corner detector [6]: $\Theta^{[n]} = -\frac{L_{vv}}{L_w} L_w{}^n = \kappa L_w{}^n$. An obvious advantage is invariance under as large a group as possible. Blom calculated n for invariance under the *affine* transformation $\begin{pmatrix} x' \\ y' \end{pmatrix} \rightarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix} (x\ y)$. The derivatives transform as $\begin{pmatrix} \partial/\partial x \\ \partial/\partial y \end{pmatrix} \rightarrow \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x'} & \frac{\partial}{\partial y'} \end{pmatrix}$. The corner detectors $\Theta^{[n]}$ transform as $\Theta^{[n]}$ = $(ad\text{-}bc)^2\{(a\ L_{x'}+c\ L_{y'})^2+(b\ L_{x'}+d\ L_{y'})^2\ \}^{(n-3)/2}\ \{2\ L_{x'}\ L_{y'}\ L_{x'y'}-L_{y'}{}^2L_{x'x'}-L_{x'}{}^2L_{y'y'}\}$.This is a relative *affine invariant* of order 2 if n=3 with the determinant $D=(ad\text{-}bc)$ of the affine transformation as order parameter. We consider here *special* affine transformations ($D=1$).

So a good corner-detector is $\Theta^{[3]} = -\frac{L_{vv}}{L_w} L_w{}^3 = L_{vv} L_w{}^2$. This feature has the nice property that is is not singular at locations where the gradient vanishes, and through its affine invariance it detects corners at all 'opening angles'.

```
im = N[Import["utrecht256.gif"][[1, 1]]];
corner1 = (gauge2D[im, 2, 0, 1] \ gauge2D[im, 0, 1, 1]^2) //
ReleaseHold;
corner2 = (gauge2D[im, 2, 0, 3] \ gauge2D[im, 0, 1, 3]^2) //
ReleaseHold; Block[{$DisplayFunction = Identity},
p1 = ListDensityPlot[im]; p2 = ListDensityPlot[corner1];
p3 = ListDensityPlot[corner2];]; Show[GraphicsArray[{p1, p2,
p3}]];
```

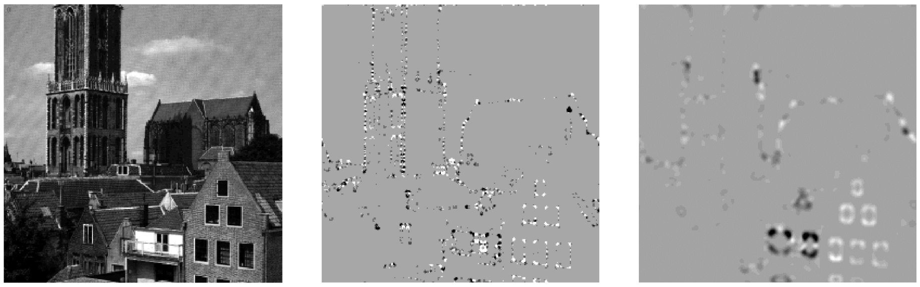


Fig. 5. a. Input image, resolution 256x256. **b.** Corner detection with $L_{vv}L_w{}^2$, σ=1 pixel. **c.** idem, σ=3 pixels.

3.4 T-junction Detection

An example of third order geometric reasoning in images is the detection of T-junctions. T-junctions in the intensity landscape of natural images occur typically at occlusion points. When we zoom in on the T-junction of an observed (i.e. blurred) image and inspect locally the isophote structure at a T-junction, we see that at a T-junction the change of the isophote curvature κ in the direction perpendicular to the isophotes (the *w*-direction) is high. So a candidate for T-junction detection is $\frac{\partial\kappa}{\partial w}$. We saw before that the isophote curvature is defined as $\kappa = -L_{vv}/L_w$. Thus the Cartesian expression for the T-junction detector becomes

```
κ= Simplify[-(dv[dv[L[x, y]]]/dw[L[x, y]]) /.
 {Lx -> D[L[x, y], x], Ly -> D[L[x, y], y]}];
τ = Simplify[dw[κ] /. {Lx -> D[L[x, y], x],
 Ly -> D[L[x, y], y]}]; % /. Derivative[n_, m_][L][x, y] ->
StringJoin[L, Table[x, {n}], Table[y, {m}]]

1/(Lx^2 + Ly^2)^3 (-Lxxy Ly^5 + Lx^4 (2Lxy^2 – Lx Lxyy + Lxx Lyy) +
Ly^4(2Lxy^2 + Lx(-Lxxx + 2Lxyy) + Lxx Lyy) + Lx^2 Ly^2 (3Lxx^2 -
Lx Lxxx - 8Lxy^2 + Lx Lxyy – 4 Lxx Lyy + 3Lyy^2) +
Lx Ly^3 (Lx Lxxy + 6 Lxx Lxy - 6Lxy Lyy – Lx Lyyy) +
Lx^3 Ly (2Lx Lxxy – 6 Lxx Lxy + 6 Lxy Lyy – Lx Lyyy))
```

To avoid singularities at vanishing gradients through the division by $(L_x^2+L_y^2)^3 = L_w^6$ we use as our T-junction detector $\tau = \frac{\partial\kappa}{\partial w} L_w{}^6$, the derivative of the curvature in the direction perpendicular to the isophotes:

```
τ= Simplify[dw[κ]\dw[L[x,y]]^6 /. {Lx→∂xL[x,y], Ly→∂yL[x, y]}];
```

Finally, we apply the T-junction detector on our blocks at a scale of σ=2:

```
τ= τ /. Derivative[n_,m_][L][x,y]→HoldForm[gD[blocks,n, m,
σ]]; σ = 2; ListDensityPlot[τ // ReleaseHold];
```

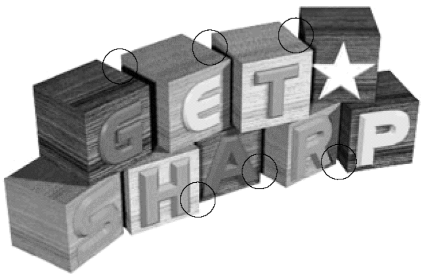
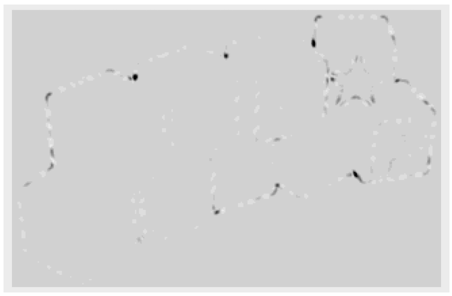


Fig. 6. a. Input image: some T-junctions encircled. Resolution 317x204 pixels.



b. T-juction detection with $\tau = \frac{\partial\kappa}{\partial w} L_w{}^6$ at a scale of 2 pixels.

3.5 Fourth Order Junction Detection

As a final fourth order example, we give an example for a detection problem in images at high order of differentiation from algebraic theory. Even at orders of differentiation as high as 4, invariant features can be constructed and calculated for discrete images through the biologically inspired scaled derivative operators. Our example is to find in a checkerboard the crossings where 4 edges meet. We take an algebraic approach, which is taken from Salden et al. [14].

When we study the fourth order local image structure, we consider the fourth order polynomial terms from the local Taylor expansion:

```
pol4=(Lxxxx x^4+4Lxxxy x^3 y+6Lxxyy x^2 y^2+4Lxyyy x y^3+Lyyyy y^4)/4!
```

The main theorem of algebra states that a polynomial is fully described by its roots: e.g. $ax^2+bx+c=(x-x_1)(x-x_2)$. Hilbert showed that the 'coincidenceness' of the roots, i.e. how well all roots coincide, is a particular invariant condition. From algebraic theory it is known that this 'coincidenceness' is given by the *discriminant*:

```
Discriminant[p_, x_] := With[{m = Exponent[p, x]},
 Cancel[((-1)^(1/2*m*(m - 1)) Resultant[p, D[p, x], x])/
 Coefficient[p, x, m]]];
```

The resultant of two polynomials *a* and *b*, both with leading coefficient one, is the product of all the differences a_i-b_j between roots of the polynomials. The resultant is always a number or a polynomial. The discriminant of a polynomial is the product of the squares of all the differences of the roots taken in pairs. We can express our function in two variables (*x*,*y*} as a function in a single variable *x/y* by the substitution $y\to 1$. Some examples:

```
Discriminant[(Lxx x^2+2 Lxy x y+Lyy y^2)/2!, x] /. {y -> 1}

Lxy^2 + Lxx Lyy
```

The discriminant of second order image structure is just the determinant of the Hessian matrix, i.e. the Gaussian curvature. Here is our fourth order discriminant:

```
Discriminant[pol4, x] /. {y -> 1}

 (497664 Lxxxy^2 Lxxyy^2 Lxyyy^2 - 31104 Lxxxx Lxxyy^3 Lxyyy^2 –
884736 Lxxxy^3 Lxyyy^3 + 62208 Lxxxx Lxxxy Lxxyy Lxyyy^3 – 648
Lxxxx^2 Lxyyy^4 - 746496 Lxxxy^2 Lxxyy^3 Lyyyy + 46656 Lxxxx Lxxyy^4
 Lyyyy + 1492992 Lxxxy^3 Lxxyy Lxyyy Lyyyy - 103680 Lxxxx Lxxxy
Lxxyy^2 Lxyyy Lyyyy - 3456 Lxxxx Lxxxy^2 Lxyyy^2 Lyyyy + 1296
Lxxxx^2 Lxxyy Lxyyy^2 Lyyyy - 373248 Lxxxy^4 Lyyyy^2 + 31104 Lxxxx
Lxxxy^2 Lxxyy Lyyyy^2 – 432 Lxxxx^2 Lxxyy^2 Lyyyy^2 - 288 Lxxxx^2
Lxxxy Lxyyy Lyyyy^2 + Lxxxx^3 Lyyyy^3)/54
```

A complicated polynomial in fourth order derivative images. Through the use of Gaussian derivative kernels each separate term can easily be calculated as an intermediate image. We change all coefficients into scaled Gaussian derivatives:

```
discr4[im_, σ_] :=
 Discriminant[pol4, x] /. {y → 1, Lxxxx → gD[im, 4, 0, σ],
 Lxxxy → gD[im, 3, 1, σ], Lxxyy → gD[im, 2, 2, σ],
 Lxyyy → gD[im, 1, 3, σ], Lyyyy → gD[im, 0, 4, σ]};
ListDensityPlot[noisycheck, ImageSize -> {200, 100}];
ListDensityPlot[discr4[noisycheck,5],ImageSize->{200, 100}];
```

The detection is rotation invariant, robust to noise, and no detection at corners:

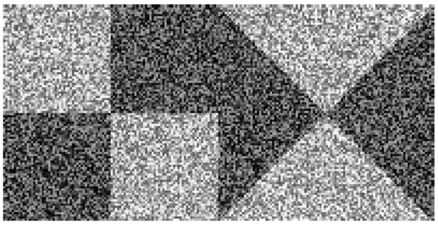
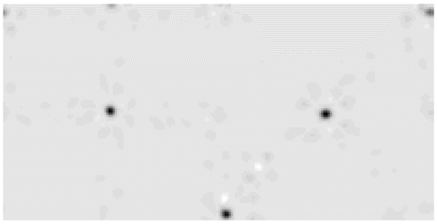


Fig. 7. a. Noisy input image. Resolution 200 x 100 pixels.


b. Four-junction detection with the algebraic discriminant D4 at σ=4 pixels.

References

1. Gregory C. DeAngelis, Izumi Ohzawa, and Ralph D. Freeman, "Receptive-field dynamics in the central visual pathways", Trends Neurosci. 18: 451-458, 1995.
2. L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, Scale and the differential structure of images, Im. and Vision Comp., vol. 10, pp. 376-388, 1992.
3. L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A Viergever, Cartesian differential invariants in scale-space, J. of Math. Im. and Vision, 3, 327-348, 1993.
4. L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, The Gaussian scale-space paradigm and the multiscale local jet, International Journal of Computer Vision, vol. 18, pp. 61-75, 1996.
5. L.M.J. Florack, Image Structure, Kluwer Ac. Publ., Dordrecht, the Nether-lands, 1997.
6. B. M. ter Haar Romeny, L. M. J. Florack, A. H. Salden, and M. A. Viergever, Higher order differential structure of images, Image & Vision Comp., vol. 12, pp. 317-325, 1994.
7. B. M. ter Haar Romeny (Ed.), Geometry Driven Diffusion in Computer Vision. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1994.
8. B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever, eds., Scale-Space '97: Proc. First Internat. Conf. on Scale-Space Theory in Computer Vision, vol. 1252 of Lecture Notes in Computer Science. Berlin: Springer Verlag, 1997.
9. B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever, Invariant third order properties of isophotes: T-junction detection, in: Theory and Applications of Image Analysis, P. Johansen and S. Olsen, eds., vol. 2 of Series in Machine Perception and Artificial Intelligence, pp. 30-37, Singapore: World Scientific, 1992.
10. T. Iijima: Basic Theory on Normalization of a Pattern - in Case of Typical 1D Pattern. Bulletin of Electrical Laboratory, vol. 26, pp. 368-388, 1962 (in Japanese).
11. J.J. Koenderink: The Structure of Images, Biol. Cybernetics, vol. 50, pp. 363-370, 1984.
12. T. Lindeberg: Scale-Space Theory in Computer Vision, Kluwer Academic Publishers, Dordrecht, Netherlands, 1994.
13. M. Nielsen, P. Johansen, O.F. Olsen, J. Weickert (Eds.), Proc. Second Intern. Conf on Scale-space Theories in Computer Vision, Lecture Notes in Computer Science, Vol. 1682, Springer, Berlin, 1999.
14. A. H. Salden, B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever, A complete and irreducible set of local orthogonally invariant features of 2-dimensional images, in: Proc. 11th IAPR Internat. Conf. on Pattern Recognition, I. T. Young, ed., The Hague, pp. 180-184, IEEE Computer Society Press, Los Alamitos, 1992.
15. J. Sporring, M. Nielsen, L. Florack: Gaussian Scale-Space Theory, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1997.

16. O. Scherzer, J. Weickert, Relations between regularization and diffusion filtering, J. Math. Imag. in Vision, in press, 2000.
17. J. Weickert, S. Ishikawa, A. Imiya, On the history of Gaussian scale-space axiomatics, in J. Sporring, M. Nielsen, L. Florack, P. Johansen (Eds.), Gaussian scale-space theory, Kluwer, Dordrecht, 45-59, 1997.
18. J. Weickert, Anisotropic diffusion in image processing, ECMI, Teubner Stuttgart, 1998.
19. R.A. Young, Simulation of Human Retinal Function with the Gaussian Derivative Model, Proc. IEEE CVPR CH2290-5, 564-569, Miami, Fla., 1986.