BASIC NUMERICAL METHODS FOR IMAGE PROCESSING

CHANG-OCK LEE

The purpose of this lecture note is to show how finite-difference schemes can be used in image analysis. We first introduce in Section 1 the main notation and consider the 2-D heat equation. The remainder of this note is concerned with the discretization of certain PDEs used in image analysis:

- Restoration by energy minimization (Section 2): We detail the discretization of the divergence term, which can also be found for the Perona and Malik equation.
- Enhancement by Osher and Rudin's shock filters (Section 3): The main interest is to use a flux limiter called minmod.
- Curve evolution with level sets and especially segmentation with geodesic active contours (Section 4). For the sake of simplicity. we examine separately each term of the model (mean curvature motion, constant speed motion, advection equation). We essentially write their discretization and give some experimental results.

For more details of finite difference methods, see [2, 6, 12]; for hyperbolic equations, see [3, 5]

Note: This lecture note is for the three hours lecture in the NIMS Workshop and Minicourse "Mathematical Analysis, Numerics and Applications in Fluid and Gas Dynamics". This is a part of the book [1].

1. Getting Started

In this lecture note we will show how certain PDEs used in image analysis can be discretized. The generic form of these PDEs is

$$\begin{cases} \mathcal{L}v = F \quad (t,x,y) \in \mathbb{R}^+ \times \Omega, \\ \dfrac{\partial v}{\partial N}(t,x,y) = 0 \quad \text{on } \mathbb{R}^+ \times \partial\Omega, \\ v(0,x,y) = f(x,y), \end{cases}$$

where Ω is the image domain and N is the normal to the boundary of Ω, denoted by $\partial\Omega$. $\mathcal{L}$ is generally a second-order differential operator such as

$$\frac{\partial v}{\partial t}(t,x,y) + H(x,y,v(t,x,y),\nabla v(t,x,y),\nabla^2 v(t,x,y)) = 0.$$

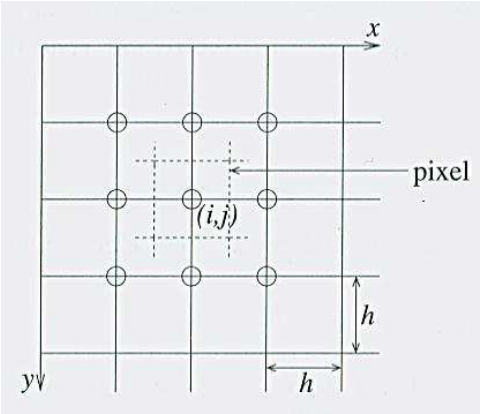

FIGURE 1: Grid on the space domain. The circles indicate the vertices that belong to the 3×3 neighborhood of the vertex (i, j).

Example: One of the simplest PDEs presented in image analysis is the heat equation:

$$\begin{cases} \dfrac{\partial v}{\partial t} = \nu \Delta v = \nu \left(\dfrac{\partial^2 v}{\partial x^2} + \dfrac{\partial^2 v}{\partial y^2} \right) & (t, x, y) \in \mathbb{R}^+ \times \Omega, \\ \dfrac{\partial v}{\partial N}(t, x, y) = 0 \quad \text{on } \mathbb{R}^+ \times \partial\Omega, \\ v(0, x, y) = f(x, y), \end{cases} \tag{1.1}$$

where ν is a positive constant.

In image processing, finite differences are widely used, which is due to the digital structure of an image as a set of pixels uniformly distributed. It is then very easy and natural to associate with an image a uniform grid, as presented in Figure 1. Since there is no reason to choose it differently, the grid spacing in the x and y directions is usually equal:

$$\Delta x = \Delta y = h.$$

Notice that in many articles from the computer vision literature, it is even choose as $h = 1$, which means that the pixel size is chosen as the unit of reference. We will call the positions (ih, jh) vertices, nodes, or pixels equivalently. We will denote by $v_{i,j}^n$ (respectively $u_{i,j}^n$) the values of the exact solution (respectively the discrete solution) at location (ih, jh) and tine $n\Delta t$.

Example: The PDE (1.1) is an initial-boundary value problem. To discretize it, we need to consider the following:

- The equation. To find the difference scheme associated with the heat equation (1.1), we use Taylor expansions expanded about the index $(n\Delta t, ih, jh)$. Naturally, the simplest method is to consider separately the discretization of each second-order derivative in x and

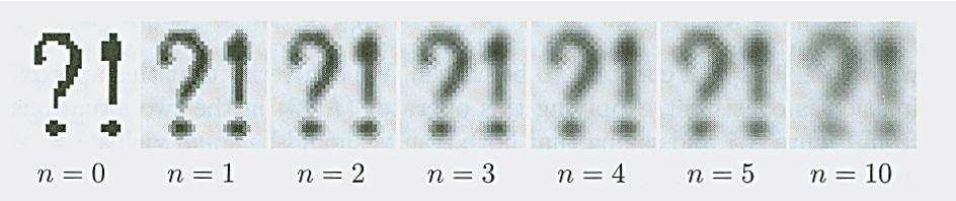

FIGURE 2: Example of results with the scheme (1.2) at different times (iterations), as applied to a simple and small image (32×32).

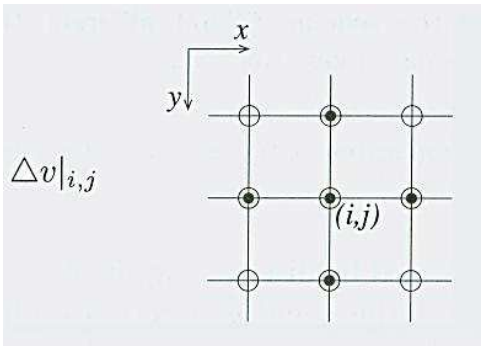

FIGURE 3: Representation of the vertices involved in the finite difference schemes. • approximation (1.3), ◯ approximation (1.4)

y. By doing so, we obtain

$$\begin{aligned} \frac{\partial v}{\partial t} - \nu \Delta v \Big|_{i,j}^{n} &= \frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} \\ &\quad - \nu \frac{v_{i+1,j}^{n} + v_{i-1,j}^{n} + v_{i,j+1}^{n} + v_{i,j-1}^{n} - 4v_{i,j}^{n}}{h^2} \\ &\quad + \mathcal{O}(\Delta t) + \mathcal{O}(h^2)\,. \end{aligned}$$

Then the difference scheme that we can propose is

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\nu \Delta t}{h^2} \left(u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n} - 4u_{i,j}^{n} \right) . \tag{1.2}$$

It is of order $(2, 1)$.

- The boundary condition. The Neumann boundary condition can be taken into account by a symmetry procedure. If the value of a pixel (vertex) that is outside the domain is needed, we use the value of the pixel that is symmetric with respect to the boundaries.
- The initial condition is simply: $u_{i,j}^{0} = g_{i,j}$, where g is the discretization of f.

To illustrate this algorithm, we show in Figure 2 some iteration as applied to a very simple image. Notice that this example shows clearly the propagation of the information as the number of iteration increases.

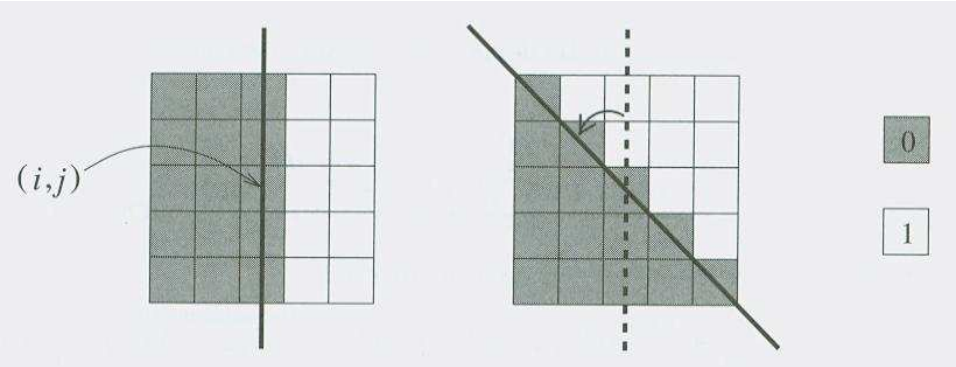

FIGURE 4: Example of a binary image representing a vertical edge and the same image after a rotation of $\pi/4$ radians. Rotationally invariant operators estimated at the point in the middle should yield the same value in both situations

Remark The scheme (1.2) was obtained by discretizing the Laplacian as a sum of the second-order derivatives in the x and y directions (see Figure 3):

$$\Delta v|_{i,j} \approx \frac{v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1} - 4v_{i,j}}{h^2}\,. \tag{1.3}$$

Clearly, this discretization does not take into account the 2-D nature and properties of this operator. To illustrate what we mean by "2-D nature and properties," we can remark that the Laplacian operator is rotationally invariant. If we apply a rotation of center (x, y) to the image v (with any angle $\theta \in [0, 2\pi)$), then $\Delta v(x, y)$ remains constant for all θ, as it should be for the discretization. Naturally, as we consider a discrete domain, we may ask only that $\Delta v|_{i,j}$ keep constant under rotations of $\pi/4$, as depicted in Figure 4. That is not the case for discretization (1.3), since we obtain 1 or 2 (for $h = 1$) depending on the situation.

To overcome this difficulty, we need to use the complete 3×3 neighborhood. We may propose the following approximation:

$$\begin{aligned}\Delta v|_{i,j} \approx\ & \lambda \frac{v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1} - 4v_{i,j}}{h^2} \\ & + (1-\lambda) \frac{v_{i+1,j+1} + v_{i-1,j+1} + v_{i+1,j-1} + v_{i-1,j-1} - 4v_{i,j}}{2h^2}\,,\end{aligned} \tag{1.4}$$

where $\lambda \in [0, 1]$ is a constant to be chosen. We can verify that this approximation is consistent. Applying this operator (1.4) in the two situations from Figure 4 and saying that both results should be equal yields $\lambda = \frac{1}{3}$, hence the approximation.

Similarly, we can propose a discretization for the first-order derivatives in x and y that is consistent with the fact that the norm of the gradient is invariant under rotation. As we will see further, a second-order centered approximation of the first derivative in x is

$$\left.\frac{\partial v}{\partial x}\right|_{i,j} \approx \delta_x v_{i,j} = \frac{v_{i+1,j} - v_{i-1,j}}{2h}\,, \tag{1.5}$$

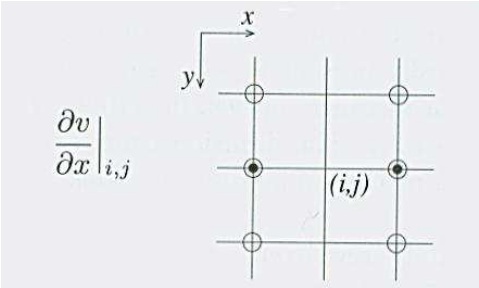

FIGURE 5: Representation of the vertices involved in the finite difference schemes. • approximation (1.5), ◯ approximation (1.6)

which can also be written in the y direction. The vertices involved in the estimation (1.5) are represented in Figure 5. As for the case of the Laplacian, these approximations are in fact "one-dimensional" and do not really take advantage of the fact that the data is of dimension 2. This is visible if we consider the value of the norm of the gradient of u in the two situations described in Figure 4: We obtain either $\frac{1}{2}$ or $\frac{1}{\sqrt{2}}$. The solution is to use more pixels in the estimation of the derivatives. In particular, we may suggest the following approximation:

$$\begin{aligned}\left.\frac{\partial v}{\partial x}\right|_{i,j} \approx{}& \lambda\frac{v_{i+1,j}-v_{i-1,j}}{2h}\\ &+\frac{(1-\lambda)}{2}\left(\frac{v_{i+1,j+1}-v_{i-1,j+1}}{2h}+\frac{v_{i+1,j-1}-v_{i-1,j-1}}{2h}\right),\end{aligned}\tag{1.6}$$

where λ is a parameter to be chosen. Applying the operator (1.6) in the two situations of Figure 5 and by saying that both results should be equal yields $\lambda=\sqrt{2}-1$, hence the approximation.

Finally we want to mention that not only is using more points in the approximations good for the rotation-invariance properties, but, practically, the result is also less sensitive to noise.The reason is that it is equivalent to perform a smoothing of the data before the estimation.

2. Image Restoration by Energy Minimization

We first consider the image restoration problem which becomes to minimize with respect to v and b the functional

$$J_\epsilon(v,b)=\frac{1}{2}\int_\Omega |Rv-v_0|^2\,dx+\lambda\int_\Omega (b|\nabla v|^2+\psi_\epsilon(b))\,dx\,.$$

The so-called half-quadratic minimization algorithm consists in minimizing successively J_ϵ with respect to each variable. The algorithm is as follows:

`For` (v^0,b^0) `given`

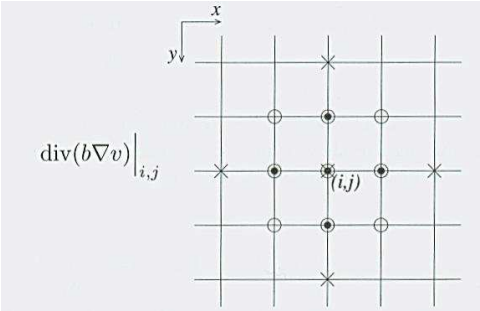

FIGURE 6: Vertices involved in the approximation of the divergence term for the different schemes. × approximation (2.1), • approximation (2.2), ◯ approximation (2.3)

- $v^{n+1} = \text{argmin}_v J_\epsilon(v, b^n)$ `i.e.,`

$$\begin{cases} R^*Rv^{n+1} - \text{div}(b^n \nabla v^{n+1}) = 0 \quad \texttt{in } \Omega, \\ b^n \dfrac{\partial v^{n+1}}{\partial N} = 0 \quad \texttt{on } \partial\Omega. \end{cases}$$

- $b^{n+1} = \text{argmin}_b J_\epsilon(v^{n+1}, b)$ `i.e.,` $b^{n+1} = \dfrac{\phi'(|\nabla v^{n+1}|)}{2|\nabla v^{n+1}|}$.
- `Go back to the first step until convergence.`

`The limit` (v^∞, b^∞) `is the solution.`

As far as discretization is concerned, the only term that may be difficult to approximate is the divergence operator. So for $b \geq 0$ and v given at nodes (i, j) the problem is to find an approximation at the node (i, j) for $\text{div}(b\nabla v)$. This kind of term is naturally present as soon as there is a regularization with a ϕ function. The diffusion operator used in the Perona and Malik model is also of the same kind.

Since this divergence operator may be rewritten as:

$$\text{div}(b\nabla v) = \frac{\partial}{\partial x}\left(b\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(b\frac{\partial v}{\partial y}\right),$$

we can use the previous one-dimensional approximation and combine them. For example, if we use the central finite-difference approximation (1.5), we have

(2.1)

$$\begin{aligned} \text{div}(b\nabla v)|_{i,j} &\approx \delta_x(b_{i,j}\delta_x v_{i,j}) + \delta_y(b_{i,j}\delta_y v_{i,j}) \\ &= \frac{1}{4h^2}\left(b_{i+1,j}v_{i+2,j} + b_{i-1,j}v_{i-2,j} + b_{i,j+1}v_{i,j+2} + b_{i,j-1}v_{i,j-2}\right. \\ &\qquad \left. - (b_{i+1,j} + b_{i-1,j} + b_{i,j+1} + b_{i,j-1})v_{i,j}\right). \end{aligned}$$

The main drawback of this representation is that it involves only the points $((i \pm 2)h, (j \pm 2)h)$, and none of the 3×3 neighborhood (see also Figure 6). This may be nonrobust for noisy data or when there is considerable variation in this region. Another possibility is to combine forward and backward

differences:

$$\begin{aligned}\operatorname{div}(b\nabla v)|_{i,j} &\approx \delta_x^+(b_{i,j}\delta_x^- v_{i,j}) + \delta_y^+(b_{i,j}\delta_y^- v_{i,j}) \\ &= \frac{1}{h^2}\left(b_{i+1,j}v_{i+1,j} + b_{i,j}v_{i-1,j} + b_{i,j+1}v_{i,j+1} + b_{i,j}v_{i,j-1}\right. \\ &\qquad \left. - (b_{i+1,j} + b_{i,j+1} + 2b_{i,j})v_{i,j}\right) .\end{aligned}$$

This approximation now involves the 3×3 neighborhood, but it introduces an asymmetry: The values of b at $((i-1)h, jh)$ and $(ih, (j-1)h)$ are not used. A solution is to use the following approximation for the derivatives:

$$\delta_x^* v_{i,j} = \frac{v_{i+\frac{1}{2},j} - v_{i-\frac{1}{2},j}}{h} \quad \text{and} \quad \delta_y^* v_{i,j} = \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{h} ,$$

where $v_{i\pm\frac{1}{2},j\pm\frac{1}{2}}$ is the value of v at location $((i \pm \frac{1}{2})h, (j \pm \frac{1}{2})h)$, which can be obtained by interpolation. As for (1.5). it is a second-order approximation. Then we have

$$\begin{aligned}\operatorname{div}(b\nabla v)|_{i,j} &\approx \delta_x^*(b_{i,j}\delta_x^* v_{i,j}) + \delta_y^*(b_{i,j}\delta_y^* v_{i,j}) \\ &= \frac{1}{h^2}\left(b_{+0}v_{i+1,j} + b_{-0}v_{i-1,j} + b_{0+}v_{i,j+1} + b_{0-}v_{i,j-1}\right. \\ &\qquad \left. - (b_{+0} + b_{-0} + b_{0+} + b_{0-})v_{i,j}\right) .\end{aligned} \tag{2.2}$$

where $b_{\pm 0} = b_{i\pm\frac{1}{2},j}$ and $b_{0\pm} = b_{i,j\mp\frac{1}{2}}$. Notice that since we applied the operators δ_x^* and δ_y^* twice, this approximation uses the values of v only at $((i \pm 1)h, (j \pm 1)h)$ (see Figure 6). However, interpolation is needed for b.

As mentioned previously for the estimation of the Laplacian, it would also be interesting to take into account the diagonal values. Then, we can look for an approximation such that

$$\begin{aligned}&\operatorname{div}(b\nabla v)|_{i,j} \approx \\ &\frac{\lambda_p}{h^2}\left(b_{+0}v_{i+1,j} + b_{-0}v_{i-1,j} + b_{0+}v_{i,j+1} + b_{0-}v_{i,j-1} - \beta_p v_{i,j}\right) \\ &+ \frac{\lambda_d}{h^2}\left(b_{++}v_{i+1,j+1} + b_{--}v_{i-1,j-1} + b_{-+}v_{i-1,j+1} + b_{+-}v_{i+1,j-1} - \beta_d v_{i,j}\right)\end{aligned} \tag{2.3}$$

with $b_{\pm\pm} = b_{i\pm\frac{1}{2},j\pm\frac{1}{2}}$,

$$\begin{cases} \beta_p = b_{0+} + b_{0-} + b_{+0} + b_{-0}, \\ \beta_d = b_{++} + b_{--} + b_{+-} + b_{-+}, \end{cases}$$

and where λ_p and λ_d are two weights to be chosen. The first condition is that the scheme must be consistent, and it can be verified that this implies

$$\lambda_p + 2\lambda_d = 1.$$

Now there remains one degree of freedom. Two possibilities can be considered:

- The first is to choose (λ_p, λ_d) constant, and for instance equal to $(\frac{1}{2}, \frac{1}{4})$, giving a privilege to the principal directions.
- The second is to choose (λ_p, λ_d) by taking into account the orientation of the gradient of v, as described in Figure 7.

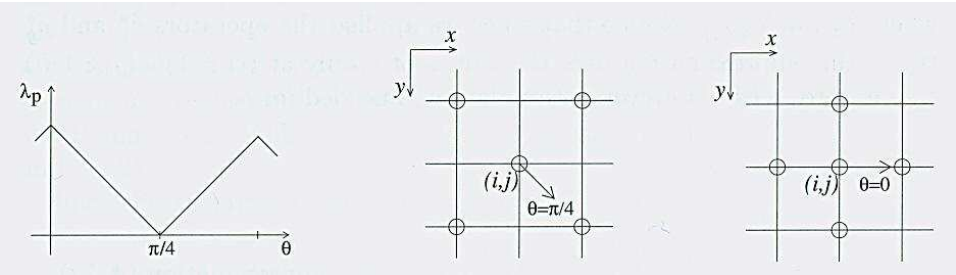

FIGURE 7: Adaptive choice of the coefficients (λ_p, λ_d) as a function of θ, the orientation of ∇v. The two right-hand figures show which points will be used in the discretization of the divergence term in two specific situations.

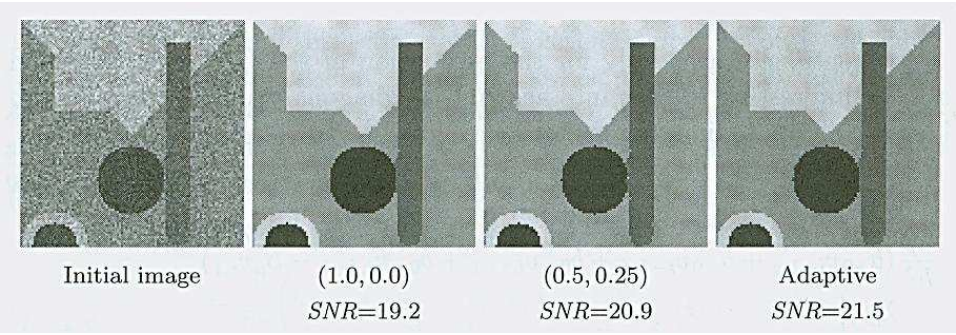

FIGURE 8: Numerical tests for the different discretization of the divergence term (the choice of (λ_p, λ_d) is indicated below the images).

We tested these different discretizations as applied to a simple image with geometric structure (see Figure 8). From left to right, an improvement in the results can be perceived (by observing the restoration of the horizontal and vertical edges). It is the adaptive choice that gives the best result.

3. Image Enhancement by Osher and Rudin Shock Filters

This section concerns the shock-filter equation proposed by Osher and Rudin [7]:

$$\frac{\partial v}{\partial t} = -|\nabla v| F(L(v)), \tag{3.1}$$

where:

- F is a Lipschitz function satisfying $F(0) = 0$, $\text{sign}(s)F(s) > 0$ $(s \neq 0)$, for example $F(s) = \text{sign}(s)$.
- L is a second-order edge detector, for example

$$L(v) = \Delta v = v_{xx} + v_{yy}$$

or

$$L(v) = \frac{1}{|\nabla v|^2}(v_x^2 v_{xx} + 2v_x v_y v_{xy} + v_y^2 v_{yy}),$$

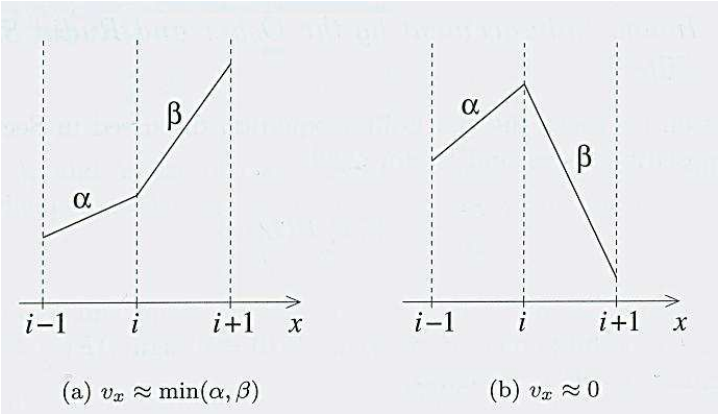

FIGURE 9: Approximation of the first derivative using the minmod function.

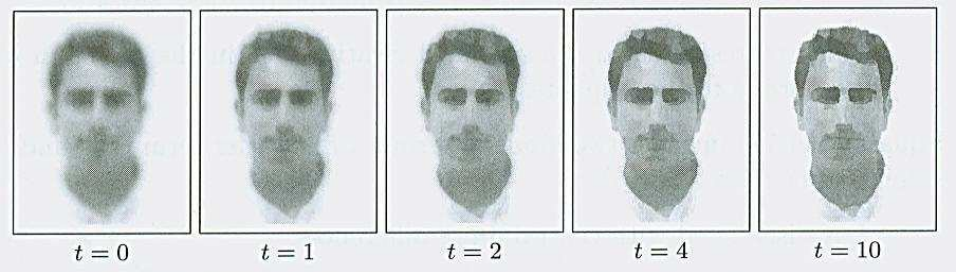

FIGURE 10: Example of the shock filter on a blurred image of a face. It shows clearly that this filter reconstructs a piecewise constant image that is not satisfying from a perceptual point of view (the result does not look like a real image).

which corresponds to the second derivative of v in the direction of the normal to the isophotes.

Equation (3.1) involves two kinds of terms: a first-order term $|\nabla v|$ and a second-order term $F(L(v))$:

- L is discretized with central finite differences.
- $|\nabla v|$ has to be approximated with more care. v_x and v_y are approximated using the minmod operator $m(\alpha, \beta)$. For instance,

$$v_x|_i = m(\delta_x^- v_i, \delta_x^+ v_i),$$

where

$$m(\alpha, \beta) = \begin{cases} \operatorname{sign}(\alpha) \min(|\alpha|, |\beta|) & \text{if} \quad \alpha\beta > 0, \\ 0 & \text{if} \quad \alpha\beta \leq 0. \end{cases}$$

This function is usually called a flux limiter. As shown in Figure 9, it permits us to choose the lowest slope, or zero in case of a local extremum (this prevents instabilities due to noise).

To summarize, the approximation of (3.1) is then given by

$$u_{i,j}^{n+1} = u_{i,j}^n - \frac{\Delta t}{h}\sqrt{(m(\delta_x^+ u_{i,j}^n, \delta_x^- u_{i,j}^n))^2 + (m(\delta_y^+ u_{i,j}^n, \delta_y^- u_{i,j}^n))^2}F_{i,j}(L(u^n)),$$

where $F_{i,j}(L(u)) = F(L_{i,j}(u))$. We show an example in Figure 10.

4. Curve Evolution with the Level-Set Method

In this section we briefly discuss the discretization of the PDEs governing curve evolution. We examine only the case where these curves are identified as level sets of the same function $u(t, x)$ (Eulerian formulation). Of course, we have in mind the geodesic active contours model given by

$$\frac{\partial v}{\partial t} = g(|\nabla I|)\,|\nabla v|\,\mathrm{div}\left(\frac{\nabla v}{|\nabla v|}\right) + \alpha g(|\nabla I|)\,|\nabla v| + \nabla g \cdot \nabla v. \tag{4.1}$$

As mentioned before, (4.1) involves two kinds of terms: a parabolic term (the first one) and hyperbolic terms (the last two). One can easily imagine that the discretization of each term needs an appropriate treatment, according to its nature (parabolic or hyperbolic). The main idea is that parabolic terms can be discretized by central finite differences, while hyperbolic terms need to be approximated by nonoscillatory upwind schemes. For the sake of clarity we start by examining the evolution driven by each of these terms.

For a detailed description of the above scheme and other numerical questions not developed here, we refer the reader to [8, 11]

4.1. **Mean Curvature Motion.** Let us consider

$$\begin{cases} \dfrac{\partial v}{\partial t} = |\nabla v|\,\mathrm{div}\left(\dfrac{\nabla v}{|\nabla v|}\right), \\ v(0, x, y) = v_0(x, y). \end{cases} \tag{4.2}$$

Equation (4.2) is a parabolic equation and has diffusive effects (like the heat equation). So, the use of upwind schemes is inappropriate, and classical central differences are used,

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t\sqrt{(\delta_x u_{i,j}^n)^2 + (\delta_y u_{i,j}^n)^2}K_{i,j}^n,$$

where $K_{i,j}^n$ is the central finite-difference approximation of the curvature:

$$K = \mathrm{div}\left(\frac{\nabla v}{|\nabla v|}\right) = \frac{v_{xx}v_y^2 + v_{yy}v_x^2 - 2v_x v_y v_{xy}}{(v_x^2 + v_y^2)^{3/2}}\,.$$

We let the reader write the expression of $K_{i,j}^n$. Unfortunately, the discretization of (4.2) is not as easy as it may appear. At some points (t, x), ∇v can be undefined, or $|\nabla v| = 0$, or $|\nabla v| = +\infty$. This situation can occur even in very simple cases [9]. For example. let us consider the shrinking of a unit circle in two dimensions which corresponds to

$$v(0, x, y) = \sqrt{x^2 + y^2} - 1,$$

i.e., v_0 is the signed distance to the unit circle. Equation (4.2) is rotationally invariant, and if we search for the solution of the form

$$v(t, x, y) = \phi(t, \sqrt{x^2 + y^2}),$$

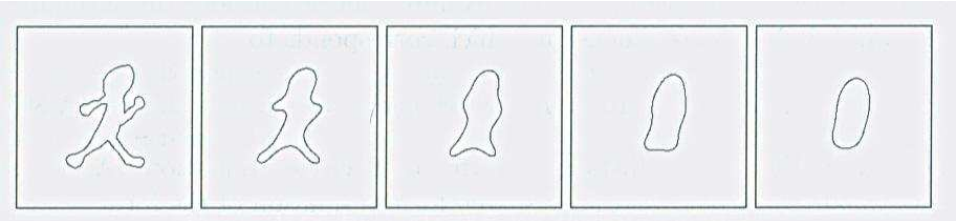

FIGURE 11: Example of mean curvature motion.

we easily get $v(t,x,y) = \sqrt{x^2+y^2+2t} - 1$, from which we deduce

$$\nabla v = \frac{1}{\sqrt{x^2+y^2+2t}} \begin{pmatrix} x \\ y \end{pmatrix}, \quad |\nabla v| = \frac{\sqrt{x^2+y^2}}{\sqrt{x^2+y^2+2t}},$$

$$\frac{\nabla v}{|\nabla v|} = \frac{1}{\sqrt{x^2+y^2}} \begin{pmatrix} x \\ y \end{pmatrix}, \quad \operatorname{div}\left(\frac{\nabla v}{|\nabla v|}\right) = \frac{1}{\sqrt{x^2+y^2}},$$

so the two last quantities are not defined at the origin, and effectively a spike occurs at the origin (see [9]). Moreover, the interface

$$\Gamma(t) = \{(x,y) : v(t,x,y) = 0\}$$

is the circle $x^2+y^2 = 1-2t$, and on $\Gamma(t)$ we have $|\nabla v|(t) = \sqrt{1-2t}$. Therefore $v(t,x,y)$ becomes more and more flat as the interface evolves and disappears at $t = \frac{1}{2}$. To circumvent this type of problem, we have to find a numerical trick that prevents the gradient norm form vanishing (or blowing up). This can be realized by reinitializing the function v from time to time to a signed distance function.

More precisely, we run (4.2) until some step n; then we solve the auxiliary PDE

$$\begin{cases} \dfrac{\partial \phi}{\partial t} + \operatorname{sign}(\phi)(|\nabla \phi| - 1) = 0, \\ \phi(0,x,y) = v(n\Delta t, x, y). \end{cases} \tag{4.3}$$

The resulting solution (as t tends to infinity), denoted by ϕ^∞, is a signed distance function whose zero-level set is the same as that of the function $v(n\Delta t, x, y)$. Then we can run again (4.2) with the initial data $v(0,x,y) = \phi^\infty(x,y)$. Practically, this reinitialization has to be done every $n = 20$ iterations of the curve evolution equation, and it is usually performed about every 5 to 10 iterations of (4.3).[1]

Remark There exists another way to avoid doing the reinitialization step. It consists in considering a modified equation (4.2) that has the property of maintaining the norm of the gradient of the solution equal to one. For further details see [4, 9].

[1]These numbers are just an indication and naturally depend on the kind of equation to be solved and on the time steps

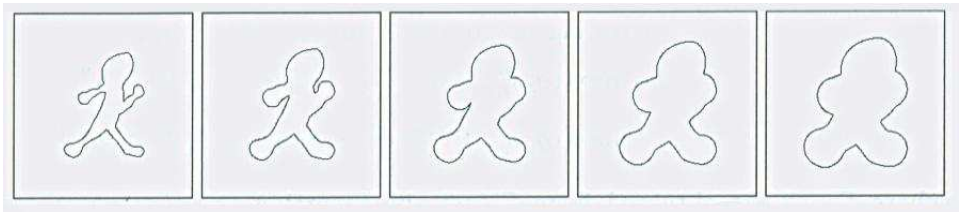

FIGURE 12: Example of constant speed motion ($c = 1$, grass fire).

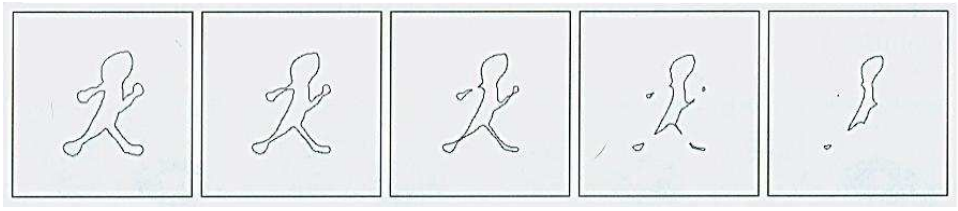

FIGURE 13: Example of constant speed motion ($c = -1$).

An example of mean curvature motion is shown in Figure 11. Notice that if we let the evolution run until convergence, any curve transforms into a circle and then collapses.

4.2. **Constant Speed Evolution.** The second example is given by

$$\begin{cases} \dfrac{\partial v}{\partial t} = c\,|\nabla v|\,, \\ v(0,x,y) = v_0(x,y), \end{cases} \tag{4.4}$$

where c is a constant. This equation describes a motion in the direction normal to the front (the corresponding Lagrangian formulation of (4.4) is $\frac{\partial \Gamma}{\partial t}(t,p) = cN(t,p)$, where N is the normal to $\Gamma(t)$). For $c = 1$, it is also referred to as *grass fire*, since it simulates a grass fire wave-front propagation.

Equation (4.4) is approximated by a nonoscillatory upwind scheme:

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t \nabla^+ u_{i,j}^n,$$

where

$$\begin{aligned} \nabla^+ u_{i,j}^n = \big[&\max(\delta_x^- u_{i,j}^n, 0)^2 + \min(\delta_x^+ u_{i,j}^n, 0)^2 \\ &+ \max(\delta_y^- u_{i,j}^n, 0)^2 + \min(\delta_y^+ u_{i,j}^n, 0)^2 \big]^{1/2} . \end{aligned}$$

We show in Figures 12 and 13 two examples of constant speed motions.

Remark Motions like equation (4.4) and more generally with a monotone speed have the following property: Every point is crossed once and only once by the curve during its evolution. Notice that this is not the case for mean curvature motion. This property can be used to derive an efficient numerical approach called the *fast marching algorithm* [10, 13]. It is beyond the scope of this lecture note to explain this method, and we refer to the original articles and to [11] for more detail.

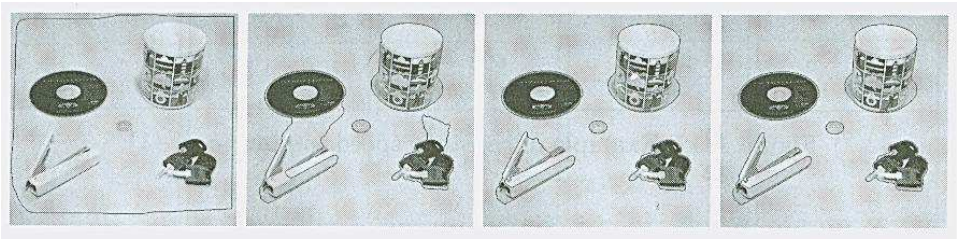

FIGURE 14: Example of segmentation. Different iterations are displayed.

4.3. **The Pure Advection Equation.** We consider here the equation

$$\begin{cases} \dfrac{\partial v}{\partial t} = A(x,y)\cdot\nabla v, \\ v(0,x,y) = v_0(x,y), \end{cases} \tag{4.5}$$

where $A(x,y) = (A_1(x,y), A_2(x,y))$. For (4.5) we use a simple upwind scheme, i.e., we check the sign of each component of A and construct a one-side upwind difference in the appropriate direction:

$$\begin{aligned} u_{i,j}^{n+1} = u_{i,j}^n + \Delta t \big[&\max((A_1)_{i,j}^n, 0)\delta_x^- u_{i,j}^n + \min((A_1)_{i,j}^n, 0)\delta_x^+ u_{i,j}^n \\ &+ \max((A_2)_{i,j}^n, 0)\delta_y^- u_{i,j}^n + \min((A_2)_{i,j}^n, 0)\delta_y^+ u_{i,j}^n\big] \ . \end{aligned}$$

4.4. **Image Segmentation by the Geodesic Active Contour Model.** Now we can consider the geodesic active contour model (4.1), which can be seen as the sum of the previous discretization. So the discrete scheme is

$$\begin{aligned} u_{i,j}^{n+1} = u_{i,j}^n + \Delta t \Big[& g_{i,j}^n K_{i,j}^n \left[(\delta_x u_{i,j}^n)^2 + (\delta_y u_{i,j}^n)^2\right]^{\frac{1}{2}} \\ &+ \alpha \left[\max(g_{i,j}^n, 0)\nabla^+ + \min(g_{i,j}^n, 0)\nabla^-\right] u_{i,j}^n \\ &+ \max((g_x)_{i,j}^n, 0)\delta_x^- u_{i,j}^n + \min((g_x)_{i,j}^n, 0)\delta_x^+ u_{i,j}^n \\ &+ \max((g_y)_{i,j}^n, 0)\delta_y^- u_{i,j}^n + \min((g_y)_{i,j}^n, 0)\delta_y^+ u_{i,j}^n\big] \ , \end{aligned}$$

where $\nabla^- u_{i,j}^n$ is obtained from $\nabla^+ u_{i,j}^n$ by inverting the signs plus and minus.

Figure 14 shows a typical example.

Remark From a numerical point of view, all the equations presented in this section involve local operations. Since we are interested only in the curve, it is enough to update the values in a band around the current position of the curve, also called a *narrow band*. Naturally, this region (band) has to be updated as the curve evolves. See, for instance, [11] for more details.

REFERENCES

[1] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing*, Applied Mathematical Sciences **147**, Springer, 2002.

[2] P.G. Ciarlet and J.L. Lions, *Handbook of Numerical Analyss. Vol. 1. Finite Difference Methods. Solution of Equations in* $\mathbb{R}^n$, North Holland, 1990.

[3] E. Godlewski and P.A. Raviart, *Hyperbolic Systems of Conservation Laws, Vol. 3/4 of Mathématiques et Applications*, Ellipses, 1991.

[4] J. Gomes and O. Faugeras, *Reconciling distance functions and level sets*, J. Vis. Comm. Image Rep. **11** (1990), 919–940.

[5] R.J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, Basel, 1992.
[6] K.W. Morton and D.F. Mayers, *Numerical Solution of Partial Differential Equations: An Introduction*, Second Edition, Cambridge University Press, 2005.
[7] S. Osher and L.I. Rudin, *Feature-oriented image enhancement using shock filters*, SIAM J. Numer. Anal. **27** (2000), 209–223.
[8] S. Osher and J. Sethian, *Fronts propagating with curvature dependent speed: algprithms based on the Hamilton-Jacobi formulation*, J. Comput. Physics. **79** (1988), 12–49.
[9] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, *A PDE-based fast local level set method*, J. Comput. Physics. **155** (1999), 420–438.
[10] J.A. Sethian, *A fast marching level set method for monotonically advancing fronts*, In Proceedings of the National Academy of Sciences. **93** (1996), 1592–1694.
[11] J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Sciences*, Cambridge Monogarphs on Applied and Computational Mathematics. Cambridge Univeristy Press, 1999.
[12] J.W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods*, Springer-Verlag, 1995.
[13] J.N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Tran. Auto. Control. **40** (1995), 1528–1538.

Department Mathematical Sciences, KAIST, Daejeon, 305-701 Korea
E-mail address: colee@kaist.edu