INF 5300 – Flexible shape extraction II

Anne Solberg (anne@ifi.uio.no)

- The pratical part of the Kass snake algorithm
- Matlab implementation
- Variations of the basic snakes
- The capture range problem
    - Distance measure based solutions
    - Gradient vector flow field based solutions

Curriculum from Chapter 6 in Nixon and Aguado

- 6.1-6.3.5
- The remaining sections can be read/studied only coarsely.

From last lecture: The energy function

- Simple snake with only two terms (no termination energy):

$$E_{snake}(s) = E_{\text{int}}(v_s) + E_{image}(v_s)$$
$$= \alpha\left|\frac{dv_s}{ds}\right|^2 + \beta\left|\frac{d^2v_s}{ds^2}\right|^2 + \gamma E_{edge}$$

- We need to approximate both the first derivative and the second derivative of v_s, and specify how E_{edge} will be computed.
- The snakes is initialized from a starting position, e.g. a circle with given center and radius.
- How should the snake iterate from its initial position?

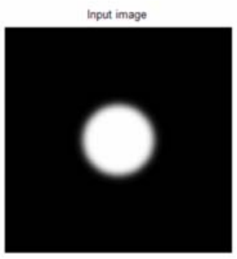
A simple image term

$$E_{image} = \int_0^1 P(v(s))ds$$

- A common way of defining P(x,y) is:

$$P(x,y) = -c\left|\nabla(G_\sigma * I(x,y))\right|$$

- c is a constant, ∇ is a gradient operator, G_σ is a Gaussian filter, and I(x,y) the input image. Note the minus sign as the gradient is high for edges.



From last lecture

- We have two equations

  $Ax=f_x(x,y)$

  $Ay=f_y(x,y)$
- These means that the snake energy should be balanced by the edge energy.
- We need an iterative approach to get a solution that is globally optimal (one single iteration by computing A^{-1} gives a local optimal solution).
- An iterative solution must have snake points that depend on time, a snake that can move.
- Let $x^{<i>},y^{<i>}$ denote the solution at time i.

The Kass snake algorithm

- Initialize the snake by selecting an initial countour
- Compute the initial energy terms and the gradient.
- Select parameters
- Given the solution at iteration i $x^{<i>},y^{<i>}$, compute $x^{<i+1>},y^{<i+1>}$:

$$x^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda x^{<i>} + f_x(x^{<i>}, y^{<i>})\right)$$
$$y^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda y^{<i>} + f_y(x^{<i>}, y^{<i>})\right)$$

The Kass differential equations

- The coordinates of the snake should be found by solving the differential equations iteratively:

$$-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\}+\frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\}+\frac{1}{2}\int_{s=0}^{1}\left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}}=0$$

$$-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{y}(s)}{ds}\right\}+\frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{y}(s)}{ds^2}\right\}+\frac{1}{2}\int_{s=0}^{1}\left.\frac{\partial E_{edge}}{\partial y}\right|_{\hat{x},\hat{y}}=0$$

- The iterative solution was given by

$$x^{<i+1>}=(A+\lambda I)^{-1}\left(\lambda x^{<i>}+f_x(x^{<i>},y^{<i>})\right)$$
$$y^{<i+1>}=(A+\lambda I)^{-1}\left(\lambda y^{<i>}+f_y(x^{<i>},y^{<i>})\right)$$

- λ is a step size

- We can write this on the form

$$f_s=a_s x_{s-2}+b_s x_{s-1}+c_s x_s+d_s x_{s+1}+e_s x_{s+2}$$

where

$$f_s=-\frac{1}{2}\left.\frac{\partial E_{edge}}{dx}\right|_{x_s,y_s}\qquad a_s=\frac{\beta_{s-1}}{h^4}\qquad b_s=-\frac{2(\beta_s+\beta_{s-1})}{h^4}-\frac{\alpha_s}{h^2}$$

$$c_s=\frac{\beta_{s+1}+4\beta_s+\beta_{s-1}}{h^4}+\frac{\alpha_{s+1}+\alpha_s}{h^2}\qquad d_s=-\frac{2(\beta_{s+1}+\beta_s)}{h^4}-\frac{\alpha_{s+1}}{h^2}\qquad e_s=\frac{\beta_{s+1}}{h_4}$$

- These are matrix equations where:

$$A=\begin{bmatrix} c_1 & d_1 & e_1 & 0 & .. & a_1 & b_1 \\ b_2 & c_2 & d_2 & e_2 & 0 & .. & a_2 \\ a_3 & b_3 & c_3 & d_3 & e_3 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & & & \\ e_{s-1} & 0 & .. & a_{s-1} & b_{s-1} & c_{s-1} & d_{s-1} \\ d_s & e_s & 0 & .. & a_s & b_s & c_s \end{bmatrix}$$

Capture range problems

- The snake must be initialized fairly close to the final target in order to get convergence.
- To make a really good initialization we need to have a very good estimate of the solution before starting the iterative process of adapting the snake.
- So we can find a good solution if we already know the solution. Obviously not very intersting...

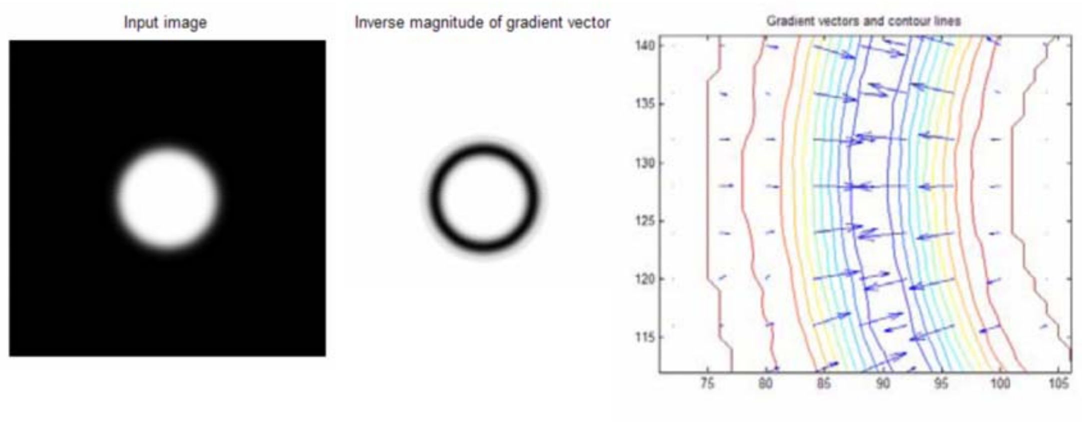
Capture range problems

- The problem stems from the short ”range” of the external fources.
- The inverse magnitude of the gradient will have significant values only in the vicinity of the salient edges.
- This basically forces us to initialize the snake very close to the target contour.
- This problem is know as the **capture range problem.**

Capture range problems

- One good way of visualizing this is by looking at the negative of the gradient of the external force field.
- These are the fources that pull the snake.
- The next slide shows this for a circle.
- Notice that outside the area in the immediate vicinity of the circle, these forces are negligible.

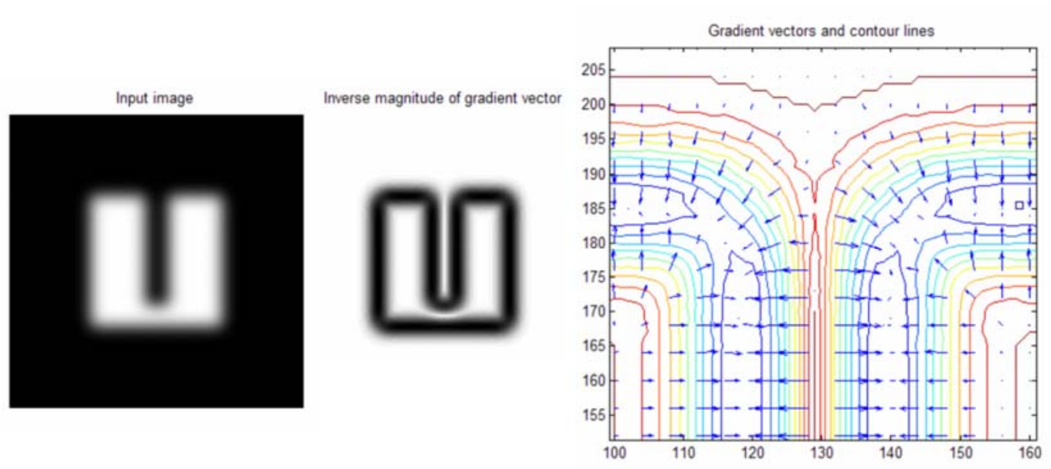
Capture range problems



Capture range problems

- This phenomenon is also the reason why you will not get convergence ito concavities, there are simply no forces to ”drag” the snake into the concavity.

Capture range problems



Capture range problems

- Many authors have suggested different methods for increasing the capture range of the external gradient vector field.
- We will look at a method suggested by Xu and Prince in: Snakes, Shapes and Gradient Vector Flow, IEEE Tr. Image Processing, vol. 7, no. 3, pp. 359-369, 1998.

Capture range problems

- Xu and Prince observed that smoothing will increase the capture range, but will NOT provide convergence into concavities.
- Other methods, for instance those based on distance maps are even better at increasing capture range, but the concavity problem remains the same.
- Xu and Prince’s solution is based on diffusing the gradient information.

Capture range problems

- Remember the snake differential equations:

$$-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\}+\frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\}+\frac{1}{2}\int_{s=0}^{1}\left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}}$$

$$\Updownarrow$$

$$-\alpha(s)\frac{d^2\hat{x}(s)}{ds^2}+\beta(s)\frac{d^4\hat{x}(s)}{ds^4}=-\frac{1}{2}\int_{s=0}^{1}\left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}}$$

- The right hand side is the negative gradient of the external force field. The limited reach of this field is our problem.
- Xu and Prince's method consists of replacing this vector field with another one – a diffused version of the gradient field.

Capture range problems

- Xu and Prince define the vector field:

$$\mathbf{v}(x,y)=(u(x,y),v(x,y))^T$$

- It is **v** that will be the GVF.
- The field **v** is the field that minimizes the following functional:

$$G=\iint \mu\left(u_x^{\;2}+u_y^2+v_x^2+v_y^2\right)+\left|\nabla f\right|^2\left|v-\nabla f\right|^2 dxdy$$

- v(x,y) is found by solving this equation.
- μ is a parameter that controls the amount of smoothing.

- Where have you seen the first term before (in this course)?

Capture range problems

$$G = \iint \mu\left(u_x^{\,2} + u_y^2 + v_x^2 + v_y^2\right) + |\nabla f|^2 |v - \nabla f|^2 \, dxdy$$

- The goal is to minimize G.
- The second term will have a minimum if v=∇f.
- If ∇f is small, the first term will dominate.
- This can also be written as

$$\mu\nabla^2 u - \left(u - f_x\right)\left(f_x^2 + f_y^2\right) = 0$$
$$\mu\nabla^2 v - \left(v - f_y\right)\left(f_x^2 + f_y^2\right) = 0$$

- If ∇f is small, what remains is Lagrange's equation:

$$\mu\nabla^2 u = 0$$
$$\mu\nabla^2 v = 0$$

Capture range problems

$$\mu\nabla^2 u = 0$$
$$\mu\nabla^2 v = 0$$

- The first term is Lagrange's equation which appear in often in physics, e.g. in heat flow or fluid flow.
- Imagine that a set of heaters is initialized at certain boundary conditions. As time evolves, the heat will redistribute/diffuse until we reach an equilibrium.
- In our setting, the gradient term act as the starting conditions.
- As the differential equation iterate, the gradient will diffuse gradually to other parts of the image in a smooth manner.

Capture range problems

- The first term will smooth the data, that is, far from edges the field will be kept as smooth as possible by imposing that the spatial derivatives be as small as possible.
- When $|\nabla f|$ is small, the vector field will be dominated by the partial derivatives of the vector field, yielding a smooth field.
- Close to edges (where $|\nabla f|$ is large) the field is forced to resemble the gradient of f itself.
- So **v** is smooth far from edges and nearly equal to the gradient of f close to edges.
- The term μ just defines the weight we give the different terms in the functional.
- The field **v** is computed iteratively

Capture range problems

- This equation has a similar solution to the original differential equation.
- We treat u and v as functions of time and solve the equations iteratively.
  - Comparable to how we iteratively computed $x^{<i+1>}, y^{<i+1>}$ from $x^{<i>}, y^{<i>}$
- The solution is obviously a numerical one, we use two sets of iterations, one for u and one for v.
- After we have computed v(x,y), we replace E_{ext} (the edge magnitude term) by v(x,y)
- So an interative algorithm is first used to compute v(x,y)

Capture range problems

- Now back to the previous image of the ”u”-shaped structure.
- This time we get driving fources that will provide convergence into concavities.

Computing **v**(x,y)

- We get a set of partial differential equations for u(x,y) and v(x,y):

$$u_t(x,y,t) = \mu\nabla^2 u(x,y,t) - \left[u(x,y,t) - f_x(x,y)\right]\left[f_x(x,y)^2 + f_y(x,y)^2\right]$$
$$v_t(x,y,t) = \mu\nabla^2 v(x,y,t) - \left[v(x,y,t) - f_y(x,y)\right]\left[f_x(x,y)^2 + f_y(x,y)^2\right]$$

- We do not go into details on how to solve this, but Xu and Prince show that they can be solved.
- Rewrite the equation

$$u_t(x,y,t) = \mu\nabla^2 u(x,y,t) - b(x,y)u(x,y,t) + c^1(x,y)$$
$$v_t(x,y,t) = \mu\nabla^2 v(x,y,t) - b(x,y)v(x,y,t) + c^2(x,y)$$
$$b(x,y) = f_x(x,y)^2 + f_y(x,y)^2 \quad \text{The gradient magnitude}$$
$$c^1(x,y) = b(x,y)f_x(x,y) \quad \text{The horisontal gradient}$$
$$c^2(x,y) = b(x,y)f_y(x,y) \quad \text{The vertical gradient}$$

Computing **v**(x,y) continued..

- Select a time step Δt and a pixel spacing Δx and Δy for the iterations.
- Approximate the partial derivatives as

$$u_t = \frac{1}{\Delta t}\left(u_{i,j}^{n+1} - u_{i,j}^{n}\right)$$
$$v_t = \frac{1}{\Delta t}\left(v_{i,j}^{n+1} - v_{i,j}^{n}\right)$$
$$\nabla^2 u = \frac{1}{\Delta x \Delta y}\left(u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{i,j}\right) \text{A Laplacian approximation}$$
$$\nabla^2 v = \frac{1}{\Delta x \Delta y}\left(v_{i+1,j} + v_{i,j+1} + v_{i-1,j} + v_{i,j-1} - 4v_{i,j}\right)$$

- Then the iterative equations are:

$$u_{i,j}^{n+1} = \left(1 - b_{i,j}\Delta t\right)u_{i,j}^{n} + r\left(u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{i,j}\right) + c_{i,j}^{1}\Delta t$$
$$v_{i,j}^{n+1} = \left(1 - b_{i,j}\Delta t\right)v_{i,j}^{n} + r\left(v_{i+1,j} + v_{i,j+1} + v_{i-1,j} + v_{i,j-1} - 4v_{i,j}\right) + c_{i,j}^{2}\Delta t$$
$$r = \frac{\mu \Delta t}{\Delta x \Delta y}$$

To get convergence we must have

$$\Delta \mathrm{t} \le \frac{\Delta x \Delta y}{4\mu}$$

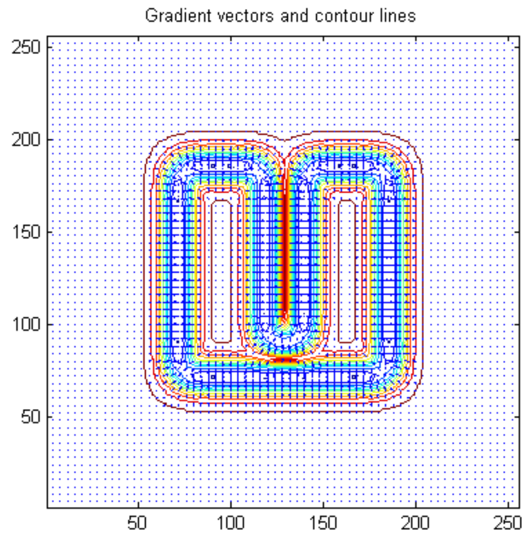
$$G = \iint \mu\left(u_x^{\,2} + u_y^2 + v_x^2 + v_y^2\right) + \left|\nabla f\right|^2 \left|v - \nabla f\right|^2 dxdy$$

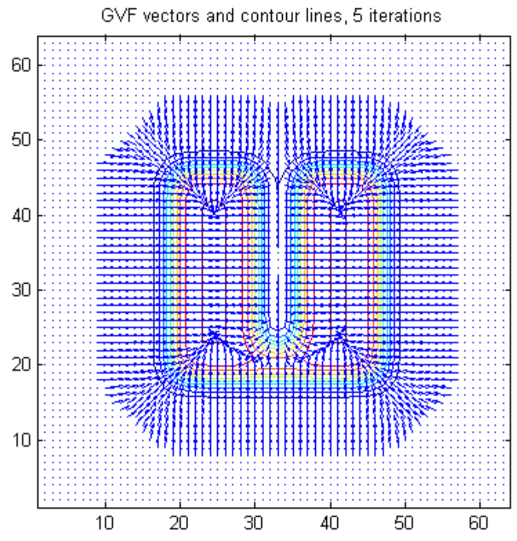
$$u_{i,j}^{n+1} = \left(1 - b_{i,j}\Delta t\right)u_{i,j}^{n} + r\left(u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{i,j}\right) + c_{i,j}^{1}\Delta t$$
$$v_{i,j}^{n+1} = \left(\underset{\substack{\uparrow\\ \text{Gradient magnitude}}}{1 - b_{i,j}\Delta t}\right)v_{i,j}^{n} + r\left(\underset{\substack{\uparrow\\ \text{Laplacian term}}}{v_{i+1,j} + v_{i,j+1} + v_{i-1,j} + v_{i,j-1} - 4v_{i,j}}\right) + \underset{\substack{\uparrow\\ \text{A term in the gradient direction}}}{c_{i,j}^{2}\Delta t}$$

Let us try to explain this….

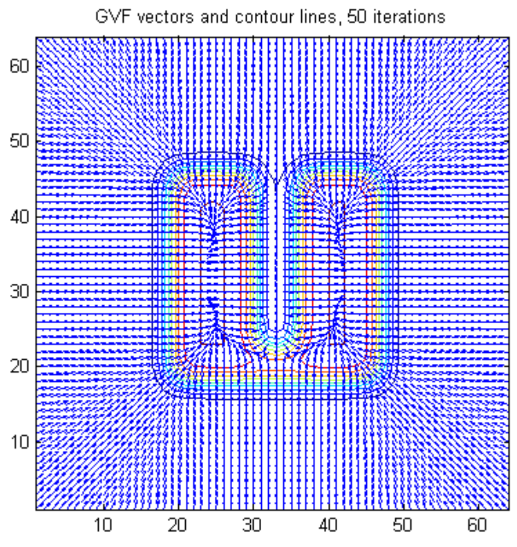
You may need to see this in high resolution in matlab

- Start with the gradient vector field and diffuse it over the image as we iterate

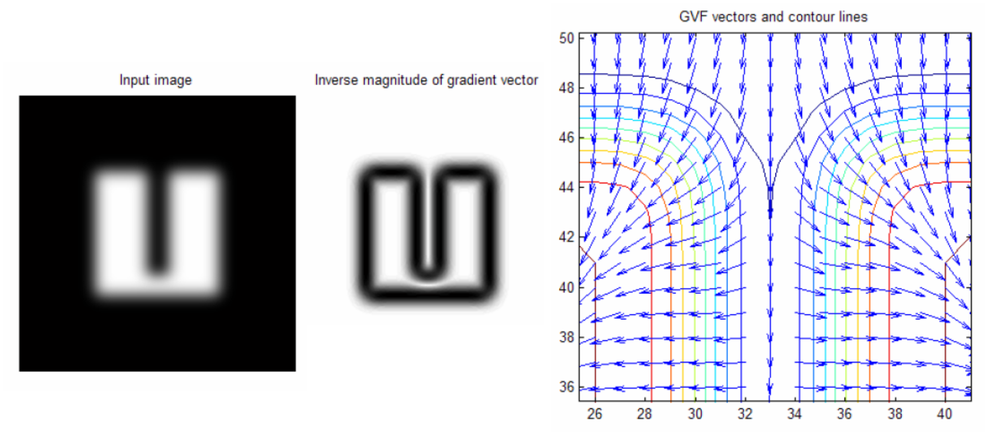




v after 50 iterations



Capture range problems


Input image
Inverse magnitude of gradient vector
GVF vectors and contour lines
50
48
46
44
42
40
38
36
26
28
30
32
34
36
38
40

Capture range problems

- Xu and Prince provide extensive material on their web address. This is an excellent site for further exploration of the GVF approach to solving thecapture range problem.
- http://iacl.ece.jhu.edu/projects/gvf/
- In order to run the GVF examples you must place all GVF matlab files provided at this address in a directory where matlab will find them.

Matlab exercise

- The Matlab scripts and images used can be found on the course web page.
- http://heim.ifi.uio.no/~inf5300/2008/defcont.zip
- They contain mostly a direct implementation of the Kass algorithm from last lecture.

Matlab demonstration

- Start by clearing the workspace and closing all windows. Load a test image.
- Uncomment the test image you want to use.

Matlab demonstration

```
% Start at scratch

clear all
close all

% Define external force field
% Read test image and convert to double

 F=imread('circ.tif','tif');
 F=double(F);

% F=imread('square.tif','tif');
% F=double(F);

%F=imread('u.tif','tif');
%F=double(F);
```

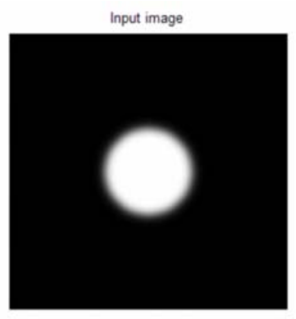
Matlab demonstration

- Then display the image, calculate the gradient images and display the horisontal and vertical gradient components.

Matlab demonstration

```
% Display it

figure
imshow(F,[]);
title('Input image');

% We want to use the negative magnitude of the gradient of this image as external
% force field so we need sobel masks

s_vert=-fspecial('sobel');
s_horz=s_vert';

% Calculate the gradient information.

F_vert=imfilter(F,s_vert,'replicate');
F_horz=imfilter(F,s_horz,'replicate');

% Lets look at these two images

figure
imshow(F_vert,[])
title('Vertical gradients')
figure
imshow(F_horz,[])
title('Horizontal gradients')
```

Matlab demonstration



What will the horisontal and vertical gradient of this look like?

Matlab demonstration

- Invert the gradient, and normalize it to have max value 1.

Matlab demonstration

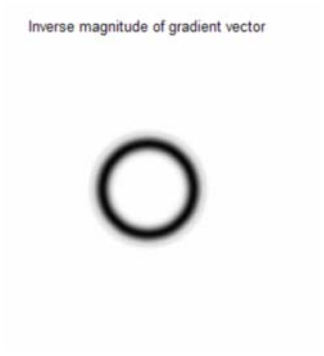
```
% Now lets calculate the negative magnitude of the gradient.
% This will be the external force field. In order to allow for
% different input images we normalize the gradient image
% to 1

P=sqrt(F_horz.*F_horz+F_vert.*F_vert);
P=P/(max(max(P)));
P=-P;
figure
imshow(P,[])
title('Inverse magnitude of gradient vector')

% Last thing, we need the two spatial derivatives
% of our external force field.  Calculate these and
% have a look at them.

P_vert=imfilter(P,s_vert,'replicate');
P_horz=imfilter(P,s_horz,'replicate');

figure
imshow(P_horz,[])
title('X derivative of force field')
figure
imshow(P_vert,[])
title('Y derivative of force field')
```

From last week: we approximated the perturbated solution for Eedge by the Taylor expansion that involved the derivates with respect to x and y

$$E_{edge}(\hat{v}(s)+\varepsilon\delta v(s)) = E_{edge}(\hat{x}(s)+\varepsilon\delta_x(s), \hat{y}(s)+\varepsilon\delta_y(s))$$

$$= E_{edge}(\hat{x}(s), \hat{y}(s)) + \varepsilon\delta_x(s)\left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}} + \varepsilon\delta_y(s)\left.\frac{\partial E_{edge}}{\partial y}\right|_{\hat{x},\hat{y}} + O(\varepsilon^2)$$

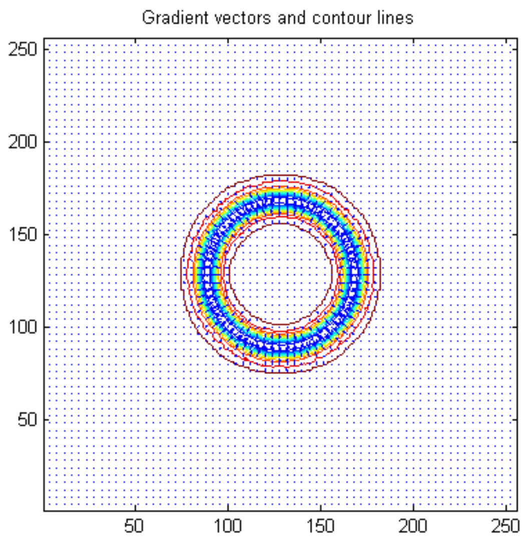
Matlab demonstration



- Display the x- and y-derivative of the force field.

Take a look at the gradient vectors

- % Lets take a look at these gradient vectors
-
- [X,Y]=meshgrid([1 4:4:256],[1 4:4:256]);
- figure
- contour(flipud(P))
- hold on
- quiver(X,flipud(Y),getmatind(-P_horz,X,Y),getmatind(P_vert,X,Y))
- axis image
- title('Gradient vectors and contour lines')



•Note that close to the border (where the gradient is non-zero), the gradient vector points in the direction of the maximum gradient.

•Acting like a force for the snake this will pull the snake in the direction of the maximum gradient

Matlab demonstration

- Then define the snake. You can vary the number of control points by setting N to different values.
- You also define an initial position for the snake, you can first make it a circle, then ”nudge” it a little.

Matlab demonstration

```
% Now lets define our snake, to begin with lets decide
% on some small number of control points (you can change
% this to your liking, the rest of the program will adapt
% gracefully)

N=20;

% Now we need to give the snake a shape.  Lets make it a circle
% and then "nudge" it a little.

x0=50*cos(0:(2*pi/(N)):(2*pi-(2*pi/(N))))+128
y0=-50*sin(0:(2*pi/(N)):(2*pi-(2*pi/(N))))+128

x0(2)=x0(2)+30;
y0(2)=y0(2)-20;
```

Matlab demonstration

- Then set the parameters for the inner forces.
- Also set the constraints that define the stiffness matrix.
- Finally, set λ.

Matlab demonstration

```
% Define the weights given to the two terms in the inner energy
% functional.  The values are NOT arbitrary.

w1=0.000001;
w2=0.01;

% Define constants for the stiffness matrix, do not edit this.

alpha=w2;
beta=-w1-4*w2;
gamma=-2*w1+6*w2;

% Define the step size

lambda = 0.2 % Stiff system
% lambda=0.1; % Unstiff system
```

Matlab demonstration

- Set up the A matrix, here we use the diag() function in matlab.

Matlab demonstration

```
% Define Stiffness matrix. The code below is just a smart way of doing
% this independently of the number of nodes.

% A =[gamma    beta     alpha    0        0     0     alpha    beta;
%    beta     gamma    beta     alpha    0     0     0        alpha;
%    alpha    beta     gamma    beta     alpha 0     0        0;
%    0        alpha    beta     gamma    beta  alpha 0        0;
%    0        0        alpha    beta     gamma beta  alpha    0;
%    0        0        0        alpha    beta  gamma beta     alpha;
%    alpha    0        0        0        alpha beta  gamma    beta;
%    beta     alpha    0        0        0     alpha beta     gamma];

A=diag(beta,-N+1)+...
  diag(alpha*ones(1,2),-N+2)+...
  diag(alpha*ones(1,N-2),-2)+...
  diag(beta*ones(1,N-1),-1)+...
  diag(gamma*ones(1,N),0)+...
  diag(beta*ones(1,N-1),+1)+...
  diag(alpha*ones(1,N-2),2)+...
  diag(alpha*ones(1,2),N-2)+...
  diag(beta,N-1)
```

Matlab demonstration

- All that remains before takeoff is to initialize x and y.
- We also set the maximum number of iterations.
- Finally, we prepare an image onto which the progress of the snake is displayed.

Matlab demonstration

```
% Initialise x and y

x=x0';
y=y0';

% The maximum number of iterations

maxiter=500;

% Weight given to external field, set to 0 or 1.

omega=1;  %How much should the gradient information be weighed?

% Display results on top of the input image

figure
imshow(P,[])
title('Snake position')
hold on
```

Matlab demonstration

- The loop is quite simple.

Matlab demonstration

```
iter=0;
while(iter<maxiter)
    c=rand(1,3); % Randomly color the snake
    %plot(x,y,'*','color',c) % Plot the snake control points
    lplot(x,y,c) % Interconnect the nodes
    iter=iter+1 % Display the iteration number
    x=(inv(A+lambda*eye(N)))*(lambda*x-
            omega*getmatind(P_horz,round(x)+1,round(y)+1));
    y=(inv(A+lambda*eye(N)))*(lambda*y-
            omega*getmatind(P_vert,round(x)+1,round(y)+1));
    dummy=input(['Press return to continue']);
end
```

Remember the equation

$$x^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda x^{<i>} + f_x(x^{<i>}, y^{<i>})\right)$$
$$y^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda y^{<i>} + f_y(x^{<i>}, y^{<i>})\right)$$

getmatind found in defcont.zip

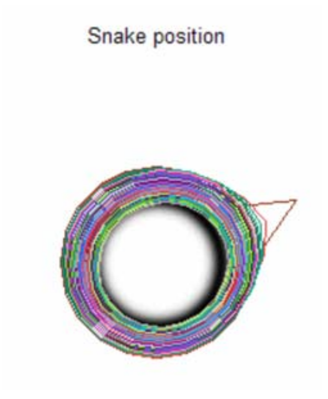
Matlab demonstration

- First, try to run the snake with only internal fources.
- Let’s first focus on the ”tension force”.
- Initialize to a circle.
- Set the weight for the external force (omega) to zero.
- What happens?

Matlab demonstration

- Take a look at the rigidity part of the internal fources. Do this by ”nudging” the contour a little.

Matlab demonstration



Matlab demonstration – try yourself

- What happens if you set w_2 to some negative value, that is, you favor contours with many ”spikes”?
- Beware: when you try this you might soon get an error saying the ”the index is out of bounds”, this is just due to the fact that this new scheme might produce x and y values that are outside the image domain. Nothing is done to handle this exception and matlab bails out.

Matlab demonstration – try yourself

- Turn the weight given to the external sources back on (omega=1).
- Run the system again.
- How do you avoid oscillations when you are close to an ideal positions?

Matlab demonstration – try yourself

- Also try this with other images like square.tif and u.tif.

- With u.tif you might observe something strange.
- What – and why?

A more difficult example

- Try the snake on this image

~inf5300/www_docs/data/seismic_timeslice.mat

The boundary we are looking for is the texture boundary between the high-frequency texture, and the homogeneous inside.
First – find a feature that has high gradient close to this boundary and low gradient elsewhere.

Use this feature image as input to the snake.
Initialize the snake using a circle both inside and outside the true contour.
Can you make it work?