Motion and Optical Flow

Monday 1 Nov 2006

video as spacetime block

- Set notation

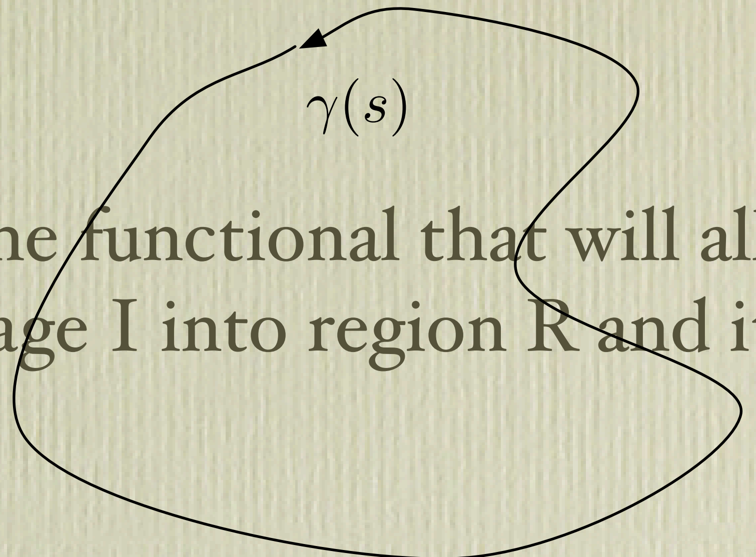
Ω is a rectangle in R^2

$$I : \Omega \times [0, \infty) \to R^+$$

vector field $x \in \Omega$

$$v(x) =< v_1(x), v_2(x) >$$

variational formulation: active contours



- We wish to define some functional that will allow us to partition the image I into region R and its complement I - R.

  Let f( ) be a monotone decreasing function, then we seek:

$$min_{\gamma} \int_{\mathcal{R}_{\gamma}} f(\delta \boldsymbol{I}(\boldsymbol{x}))d\boldsymbol{x} + \lambda \int_{\gamma} ds$$

Q. What does minimizing this functional favor?

variational approach ...

$$min_\gamma \int_{\mathcal{R}_\gamma} f(\delta \boldsymbol{I}(\boldsymbol{x}))d\boldsymbol{x} + \lambda \int_\gamma ds$$

- minimizing favors regions of large gradient of I, and at the same time controls (minimizes) length of boundary

  Minimizing not over real numbers, but over function spaces: eg over curves,
  Apply calculus of variations.

  Solving Euler-Lagrange equations for that functional yields this differential equation,
  called an evolution equation:

evolution equation

- Evolution equation for functional is an ODE to vary the boundary curve:

$$\frac{\partial \gamma}{\partial \tau} = F\vec{\nu} = (f(\delta \boldsymbol{I}(\boldsymbol{x})) + \lambda\kappa)\vec{\nu}$$

$\vec{\nu}$ inward normal to curve γ

κ geodesic curvature

As $\delta \boldsymbol{I} \to \infty,\ f(\delta \boldsymbol{I}) \to 0$

so the balloon force pushes contour to large gradient image areas

sphere inversion problem

- old and new approaches

  Thurston proof & video

  Sullivan proof & video
  using curvature-driven flow

motion estimation

- different criteria for
  compression: motion-compemsated compressopn (MPEG)
  vs
  motion-based video segmentation

  skip many apparent motion effects due to variations in illumination or camera characteristics
  focus on object-induced motion

models of motion

- spatial models
  temporal models
  region of support

spatial model: assume that the movement of a dot at position x is modeled by some affine map

$$v(x) = \begin{pmatrix} b1 \\ b2 \end{pmatrix} + \begin{pmatrix} b3 & b4 \\ b5 & b6 \end{pmatrix} x$$

temporal model of motion

temporal model
assuming velocity is constant between time t and $\tau > t$

$$\boldsymbol{x}(\tau) = \boldsymbol{x}(t) + v_t(x)(\tau - t) = \boldsymbol{x}(t) + \boldsymbol{d}_{t,\tau}(x)$$

- and ... region of support

observation models

- Key assumption: Image intensity of a (point) object does not change along motion trajectory, so , for every **x**:

$$I_k[\boldsymbol{n}] = I_{k-1}[\boldsymbol{n} - \boldsymbol{d}]$$

Differentiating w/r s, where s is length along trajectory:

$$\frac{dI}{ds} = 0$$

by chain rule:

$$\frac{dI}{dx}\nu_1 + \frac{dI}{dy}\nu_2 + \frac{dI}{dt} = (\nabla I) \cdot \boldsymbol{\nu} + \frac{dI}{dt} = 0$$

regularization of image

- Underconstrained -- not enough conditions to yield a motion. Assume neighboring points move alike. One way: <u>motion field is locally smooth</u>, with low gradient. We minimize E[**v**] for a velocity field

$$\int_D (\nabla I(x) \cdot \boldsymbol{v}(x) + \frac{\partial I(x)}{dt})^2 + \lambda(\|\nabla(v_1(x))\|^2 + \|\nabla(v_2(x))\|^2)$$

estimation criteria

- (Boldfaced are 2-vectors in $\mathbb{Z}^2$)
  d[**n**] = displaced image of point **n** under the vector field **v**[**n**] = **d**[**n**] - **n**

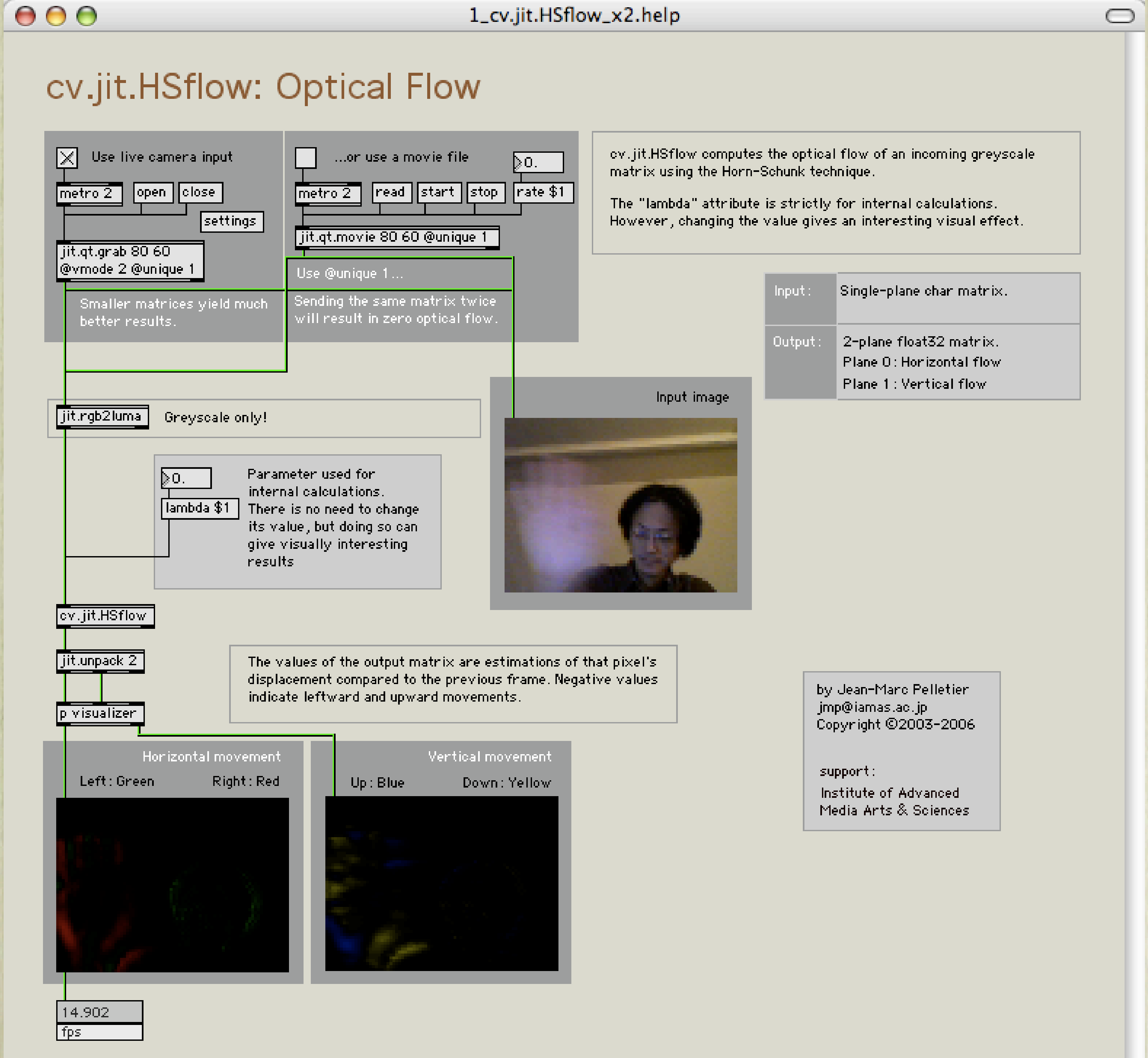
  estimated image intensity: $\tilde{I}_k[\boldsymbol{n}]$

$$\tilde{I}_k[\boldsymbol{n}] \equiv I_{k-1}[n - d[n]]$$

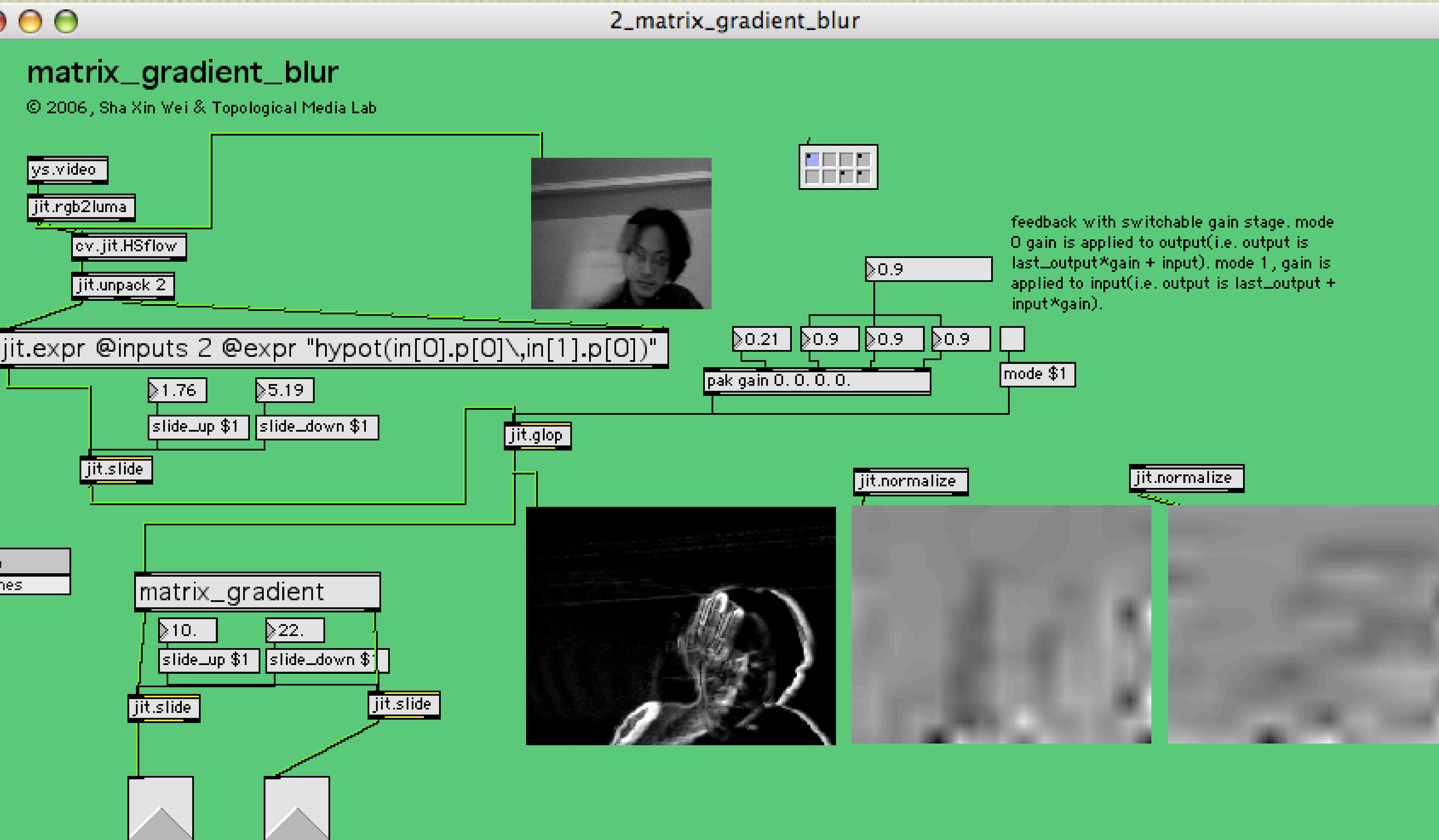
Find **d** that minimizes an error function. A reasonable one is not quadratic (too many outliers) but simply:

$$\mathcal{E}[d] = \sum_{n \in \mathcal{R}} |I_k[n] - \tilde{I}_k[\boldsymbol{n}]|$$

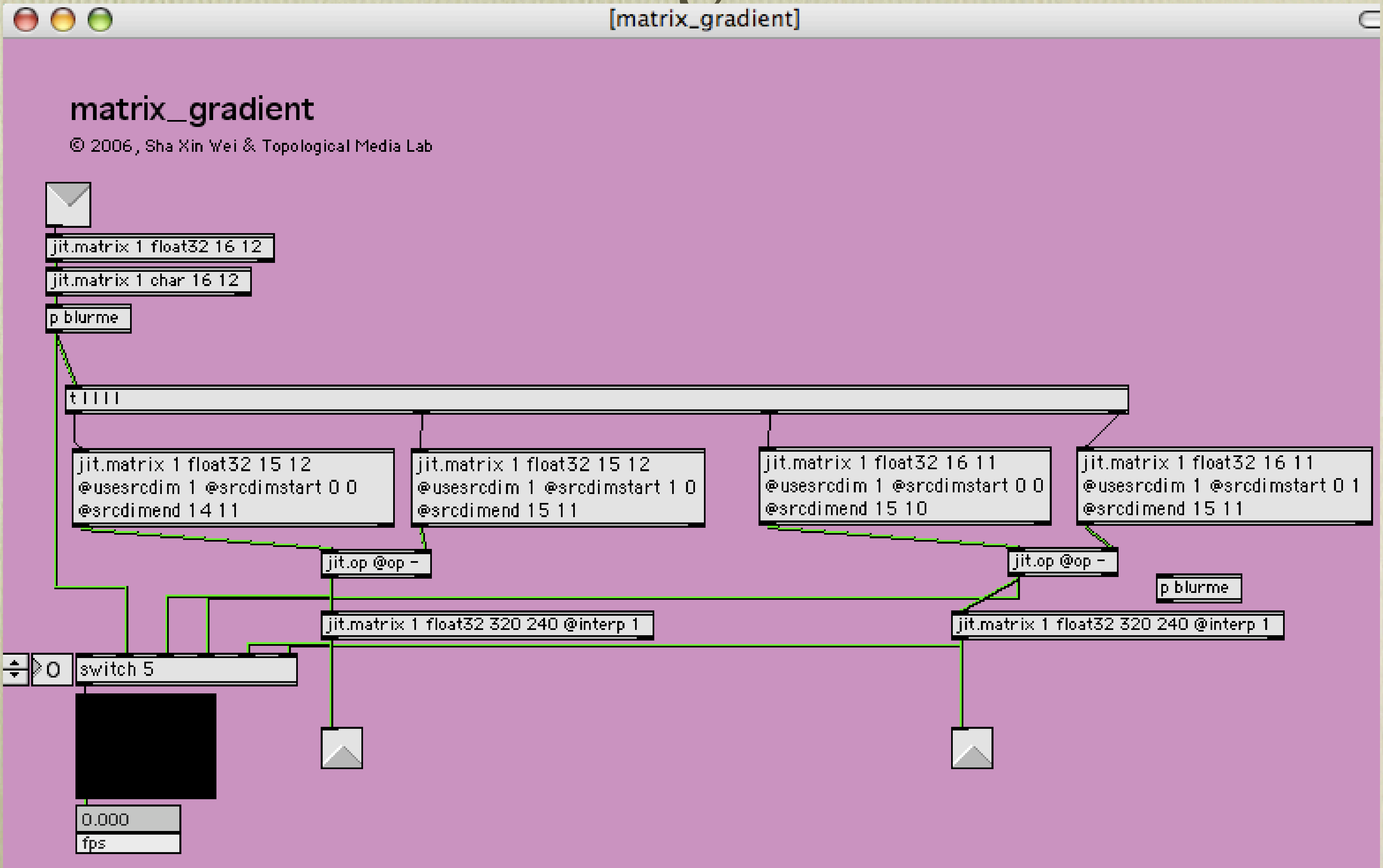
examples

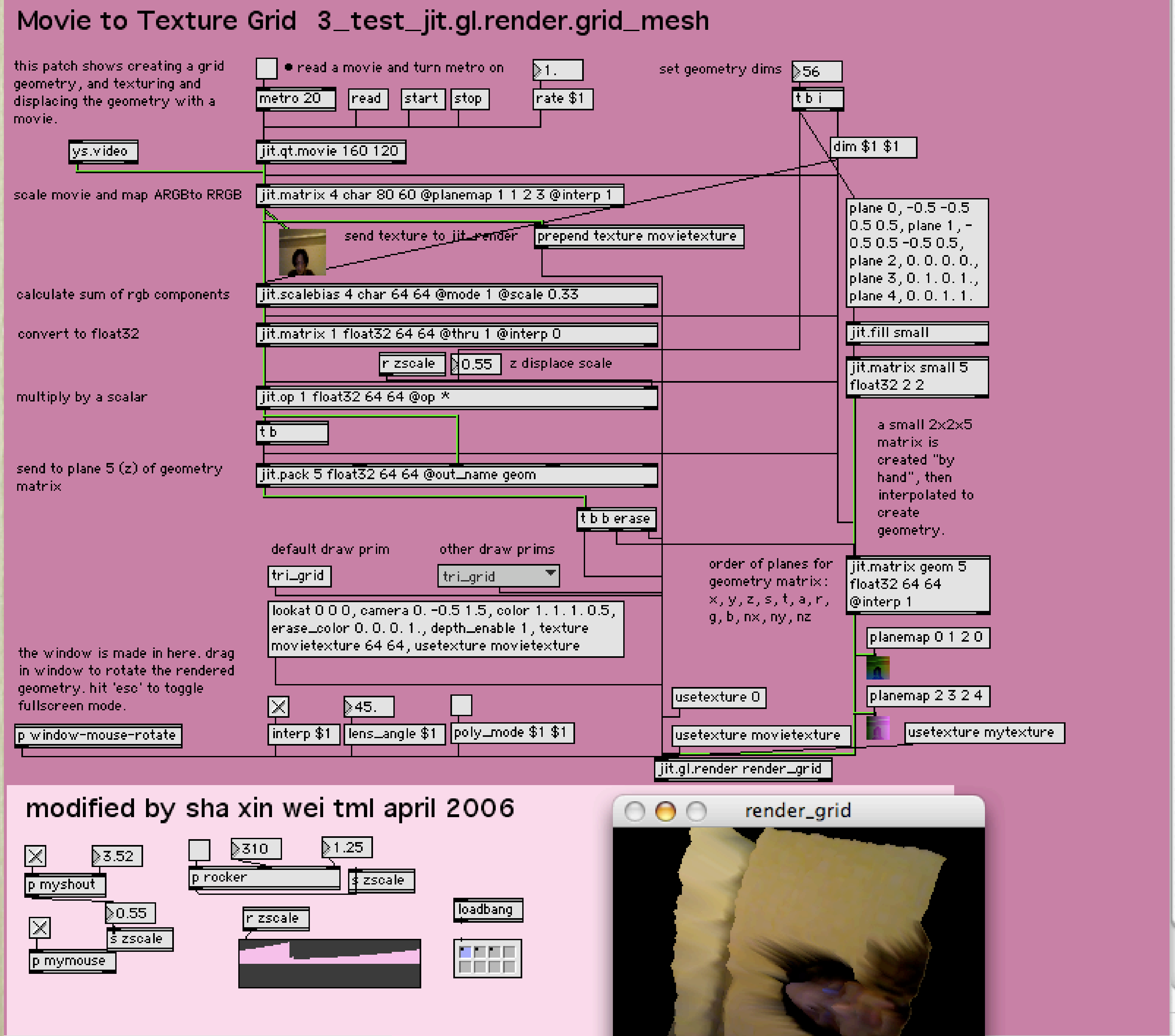

1_cv.jit.HSflow_x2.help
cv.jit.HSflow: Optical Flow
Use live camera input
metro 2
open
close
settings
jit.qt.grab 80 60 @vmode 2 @unique 1
Smaller matrices yield much better results.
...or use a movie file
0.
metro 2
read
start
stop
rate $1
jit.qt.movie 80 60 @unique 1
Use @unique 1 ...
Sending the same matrix twice will result in zero optical flow.
cv.jit.HSflow computes the optical flow of an incoming greyscale matrix using the Horn-Schunk technique.
The "lambda" attribute is strictly for internal calculations. However, changing the value gives an interesting visual effect.
Input: Single-plane char matrix.
Output: 2-plane float32 matrix. Plane 0: Horizontal flow Plane 1: Vertical flow
jit.rgb2luma
Greyscale only!
Input image
0.
lambda $1
Parameter used for internal calculations. There is no need to change its value, but doing so can give visually interesting results
cv.jit.HSflow
jit.unpack 2
p visualizer
The values of the output matrix are estimations of that pixel's displacement compared to the previous frame. Negative values indicate leftward and upward movements.
by Jean-Marc Pelletier
jmp@iamas.ac.jp
Copyright ©2003-2006
support:
Institute of Advanced Media Arts & Sciences
Horizontal movement
Left: Green
Right: Red
Vertical movement
Up: Blue
Down: Yellow
14.902
fps

matrix_gradient_blur

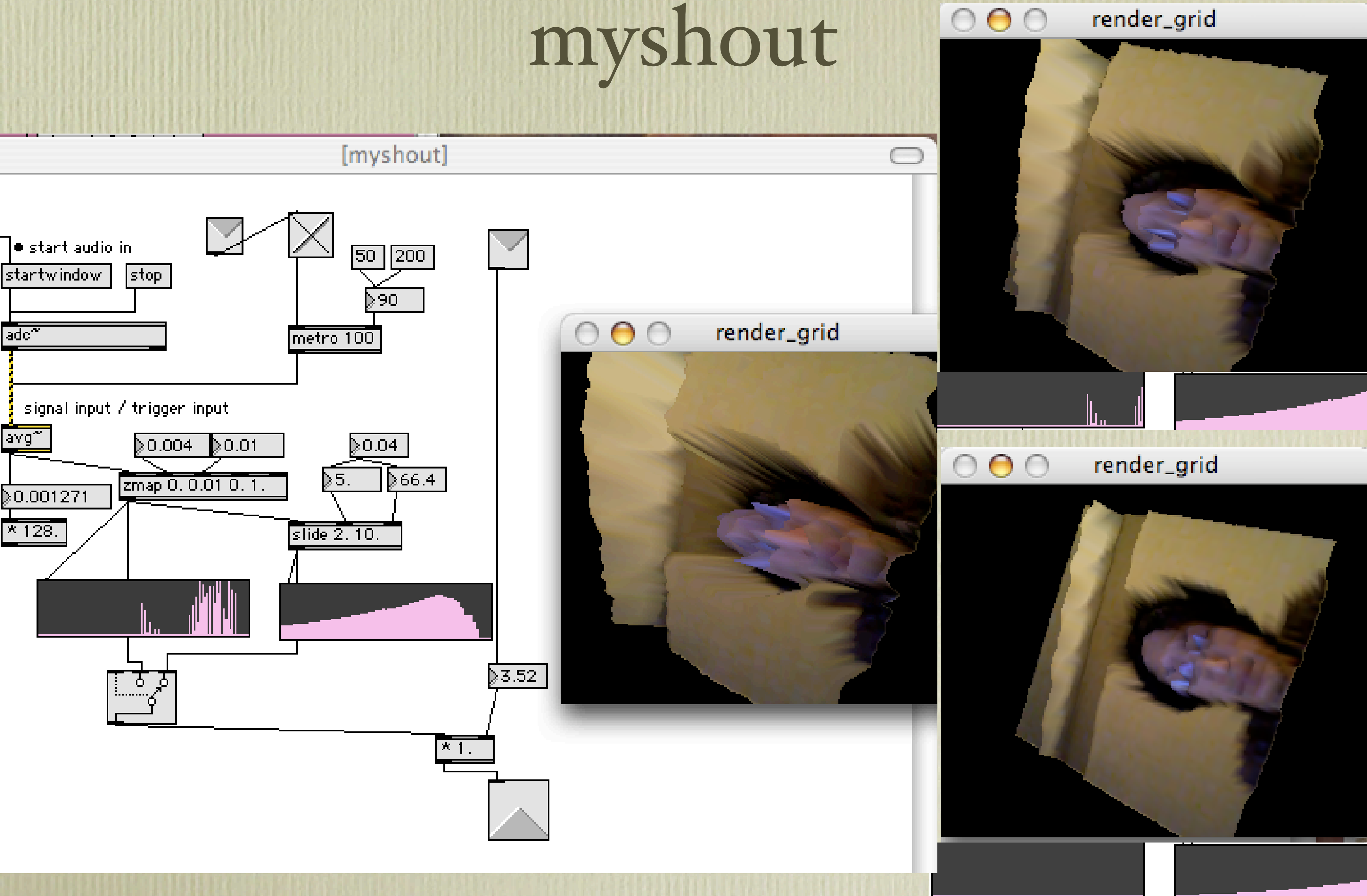


matrix_gradient

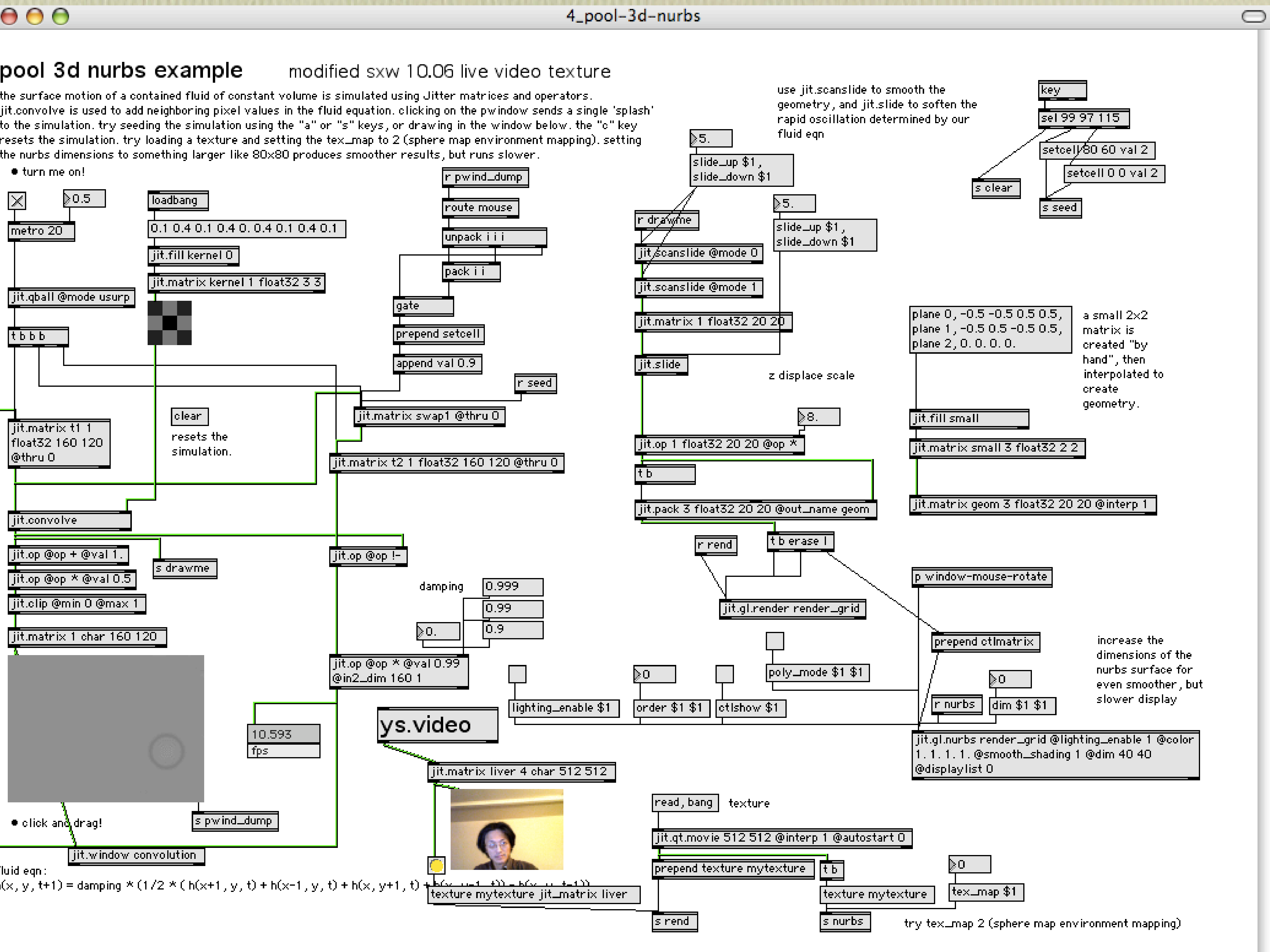



Movie to Texture Grid 3_test_jit.gl.render.grid_mesh
this patch shows creating a grid geometry, and texturing and displacing the geometry with a movie.
read a movie and turn metro on
metro 20
read
start
stop
1.
rate $1
set geometry dims
56
t b i
dim $1 $1
ys.video
jit.qt.movie 160 120
scale movie and map ARGBto RRGB
jit.matrix 4 char 80 60 @planemap 1 1 2 3 @interp 1
send texture to jit_render
prepend texture movietexture
plane 0, -0.5 -0.5 0.5 0.5, plane 1, -0.5 0.5 -0.5 0.5, plane 2, 0. 0. 0. 0., plane 3, 0. 1. 0. 1., plane 4, 0. 0. 1. 1.
calculate sum of rgb components
jit.scalebias 4 char 64 64 @mode 1 @scale 0.33
convert to float32
jit.matrix 1 float32 64 64 @thru 1 @interp 0
jit.fill small
r zscale
0.55
z displace scale
jit.matrix small 5 float32 2 2
multiply by a scalar
jit.op 1 float32 64 64 @op *
t b
a small 2x2x5 matrix is created "by hand", then interpolated to create geometry.
send to plane 5 (z) of geometry matrix
jit.pack 5 float32 64 64 @out_name geom
t b b erase
default draw prim
other draw prims
tri_grid
tri_grid
order of planes for geometry matrix: x, y, z, s, t, a, r, g, b, nx, ny, nz
jit.matrix geom 5 float32 64 64 @interp 1
lookat 0 0 0, camera 0. -0.5 1.5, color 1. 1. 1. 0.5, erase_color 0. 0. 0. 1., depth_enable 1, texture movietexture 64 64, usetexture movietexture
planemap 0 1 2 0
the window is made in here. drag in window to rotate the rendered geometry. hit 'esc' to toggle fullscreen mode.
45.
usetexture 0
planemap 2 3 2 4
p window-mouse-rotate
interp $1
lens_angle $1
poly_mode $1 $1
usetexture movietexture
usetexture mytexture
jit.gl.render render_grid
modified by sha xin wei tml april 2006
render_grid
3.52
310
1.25
p myshout
p rocker
s zscale
0.55
r zscale
loadbang
s zscale
p mymouse

myshout



[myshout]
start audio in
startwindow
stop
adc~
metro 100
50
200
90
signal input / trigger input
avg~
0.004
0.01
0.04
zmap 0. 0.01 0. 1.
5.
66.4
0.001271
* 128.
slide 2. 10.
3.52
* 1.
render_grid
render_grid
render_grid


4_pool-3d-nurbs
pool 3d nurbs example
modified sxw 10.06 live video texture
the surface motion of a contained fluid of constant volume is simulated using Jitter matrices and operators. jit.convolve is used to add neighboring pixel values in the fluid equation. clicking on the pwindow sends a single 'splash' to the simulation. try seeding the simulation using the "a" or "s" keys, or drawing in the window below. the "c" key resets the simulation. try loading a texture and setting the tex_map to 2 (sphere map environment mapping). setting the nurbs dimensions to something larger like 80x80 produces smoother results, but runs slower.
turn me on!
use jit.scanslide to smooth the geometry, and jit.slide to soften the rapid oscillation determined by our fluid eqn
key
sel 99 97 115
setcell 80 60 val 2
setcell 0 0 val 2
s clear
s seed
0.5
metro 20
loadbang
0.1 0.4 0.1 0.4 0. 0.4 0.1 0.4 0.1
jit.fill kernel 0
jit.matrix kernel 1 float32 3 3
jit.qball @mode usurp
t b b b
r pwind_dump
route mouse
unpack i i i
pack i i
gate
prepend setcell
append val 0.9
r seed
jit.matrix swap1 @thru 0
clear
resets the simulation.
jit.matrix t1 1 float32 160 120 @thru 0
jit.matrix t2 1 float32 160 120 @thru 0
jit.convolve
jit.op @op + @val 1.
s drawme
jit.op @op * @val 0.5
jit.clip @min 0 @max 1
jit.matrix 1 char 160 120
jit.op @op !-
damping
0.999
0.99
0.9
0.
jit.op @op * @val 0.99 @in2_dim 160 1
10.593
fps
click and drag!
s pwind_dump
jit.window convolution
fluid eqn:
h(x, y, t+1) = damping * (1/2 * (h(x+1, y, t) + h(x-1, y, t) + h(x, y+1, t) + h(x, y-1, t)) - h(x, y, t-1))
5.
slide_up $1, slide_down $1
r drawme
jit.scanslide @mode 0
jit.scanslide @mode 1
jit.matrix 1 float32 20 20
jit.slide
z displace scale
8.
jit.op 1 float32 20 20 @op *
t b
jit.pack 3 float32 20 20 @out_name geom
plane 0, -0.5 -0.5 0.5 0.5, plane 1, -0.5 0.5 -0.5 0.5, plane 2, 0. 0. 0. 0.
a small 2x2 matrix is created "by hand", then interpolated to create geometry.
jit.fill small
jit.matrix small 3 float32 2 2
jit.matrix geom 3 float32 20 20 @interp 1
r rend
t b erase l
jit.gl.render render_grid
p window-mouse-rotate
prepend ctlmatrix
increase the dimensions of the nurbs surface for even smoother, but slower display
lighting_enable $1
order $1 $1
ctlshow $1
poly_mode $1 $1
r nurbs
dim $1 $1
jit.gl.nurbs render_grid @lighting_enable 1 @color 1. 1. 1. 1. @smooth_shading 1 @dim 40 40 @displaylist 0
ys.video
jit.matrix liver 4 char 512 512
texture mytexture jit_matrix liver
read, bang
texture
jit.qt.movie 512 512 @interp 1 @autostart 0
prepend texture mytexture
t b
texture mytexture
tex_map $1
s rend
s nurbs
try tex_map 2 (sphere map environment mapping)

pool-3d-nurbs