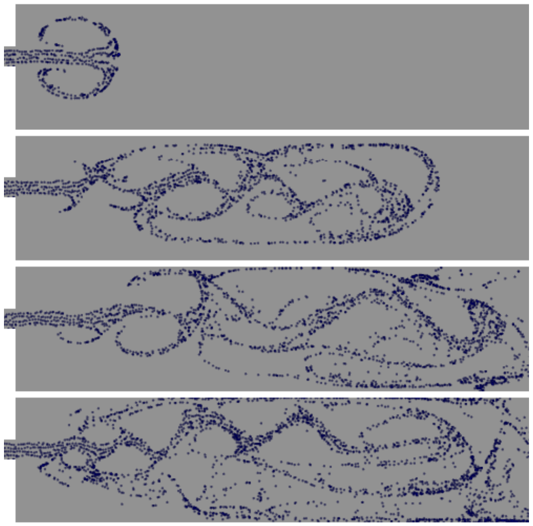
Flow Visualization

CS 176 Spring 2011

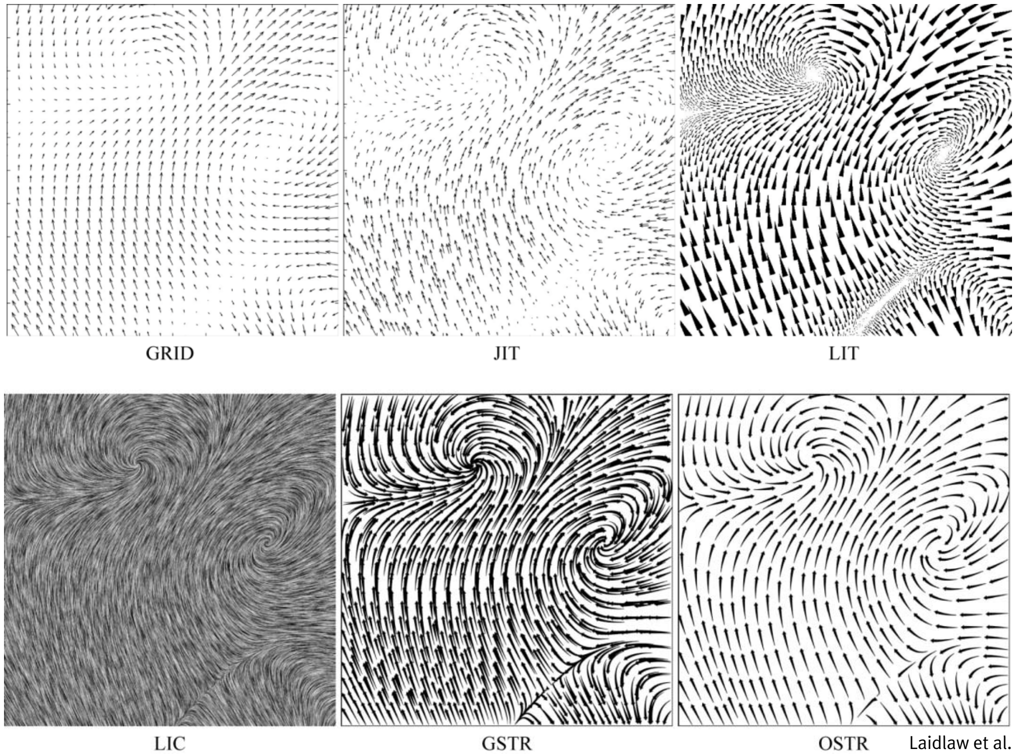
Fluid Simulation

Looking at vector fields…

- how?

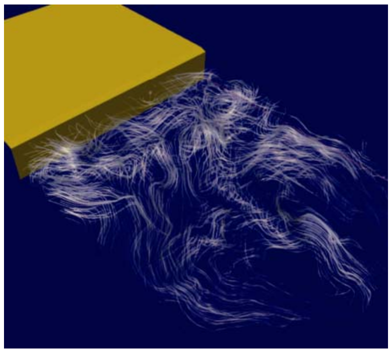


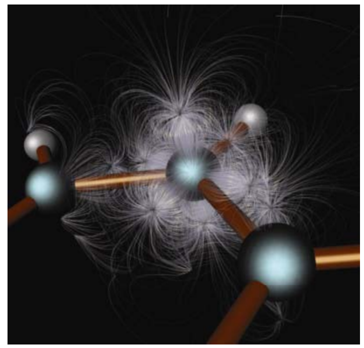
Bolz et al.

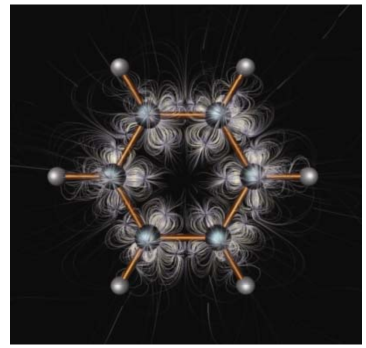


Laidlaw et al.

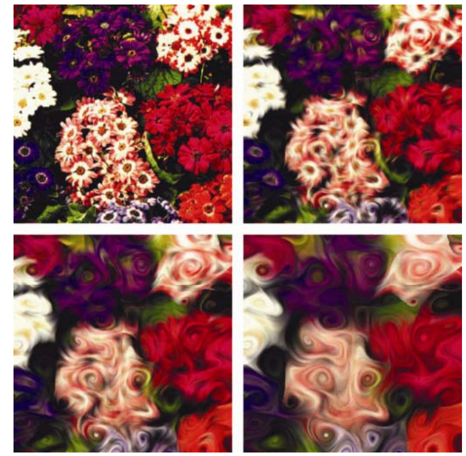
Field Visualization



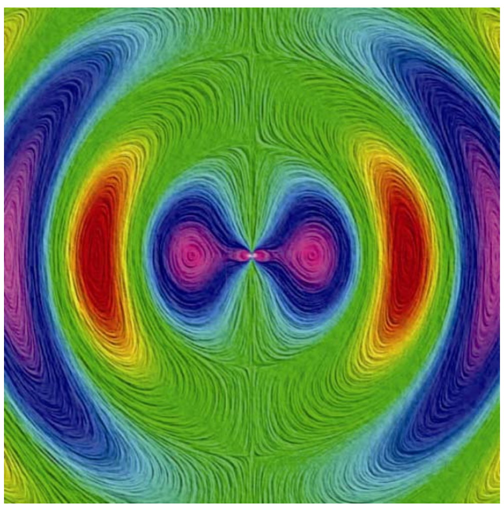


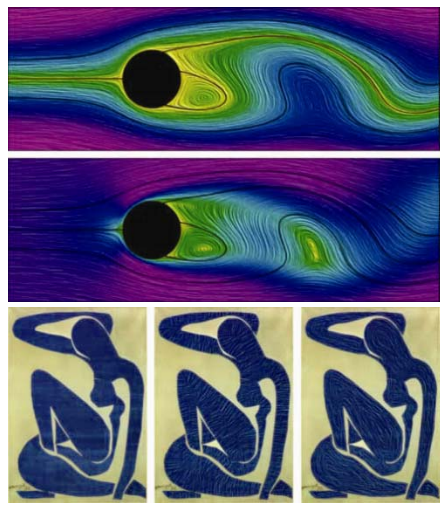


Stalling et al.



Cabral et al.



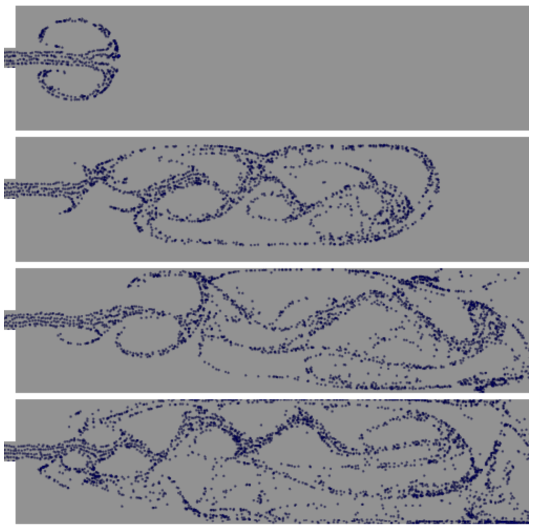


Stalling et al.

Methods

Physical analogies

- particle advection
  - static or dynamic
- ink advection



Bolz et al.

Issues:

- missing important detail

Texture Methods

Advection

- advect texture coordinates with velocity; display regular grid with original texture
  - next time step: use previous texture
  - blend with exponential decay

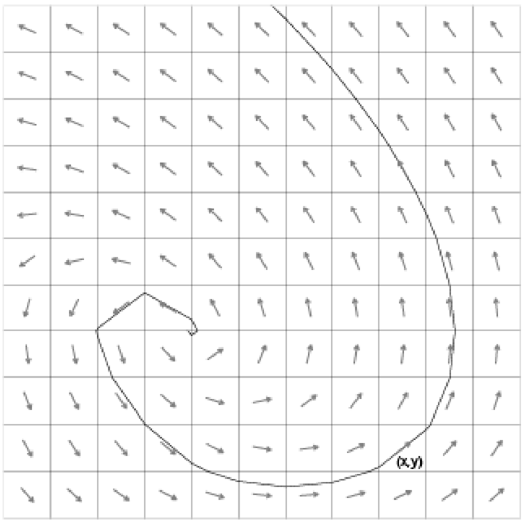
Line Integral Convltn.

Continuous version

- integral curves of the flow

$$\frac{d}{dt}\sigma(t) = v(\sigma(t))$$

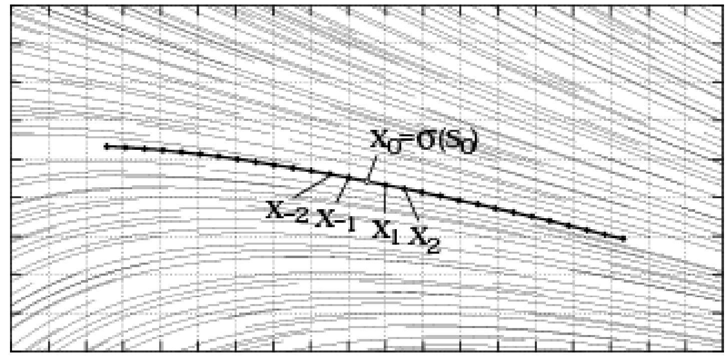
- particles in flow
- follow one traj.



Integrate Along Flow

Integrate ***noise***

- output pixel:



$$I(x_0) = \int_{s_0-L}^{s_0+L} k(s - s_0)T(\sigma(s))ds$$

noise texture

- high correlation along flow
- low correlation orthogonal to flow

Making it Fast

Pixels along path are correlated

- curve tracing for each output pixel far to expensive
  - step along curve!

$$I(x_2) = I(x_1) - \int_{s_1-L}^{s_1-L+\triangle s} T(\sigma(s))ds + \int_{s_1+L}^{s_1+L+\triangle s} T(\sigma(s))ds$$

  - accumulate results in pixels crossed
  - renormalize at end

Details

Streamline integration

- high order integrator recommended
  - may make LARGE steps
  - need to interpolate along path
- aliasing...
  - step size 1/2 of texture cell
- path length: 1/10 image size

Animation

Slide integration along streamline

- blend multiple images with different phases
  - contrast and brightness (renormalize)

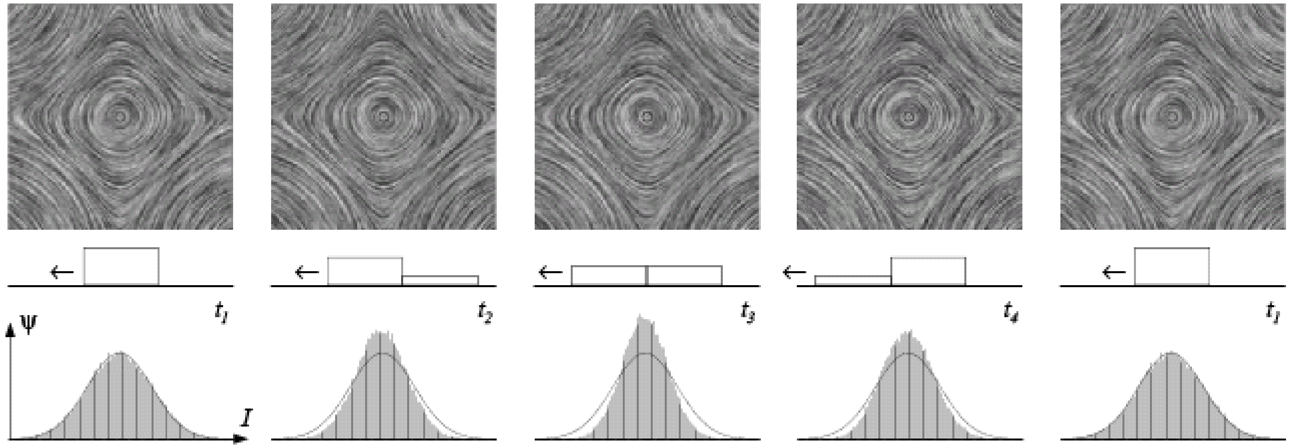


Image Based Flow Viz

Treat images as basic primitive

- LIC as blending of advected images
  - setup mesh with advected texture coordinates
  - render and blend; repeat

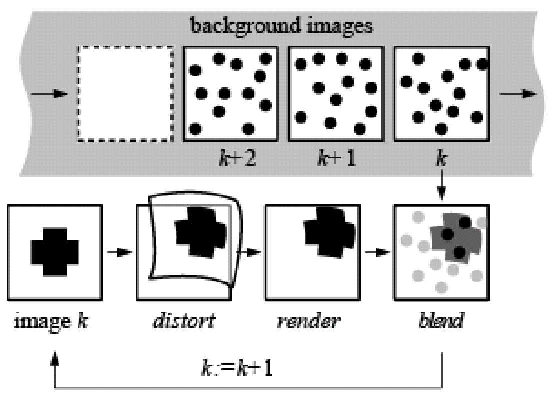
$$F(p_k;k) = (1-\alpha)F(p_{k-1};k-1) + \alpha G(p_k;k)$$

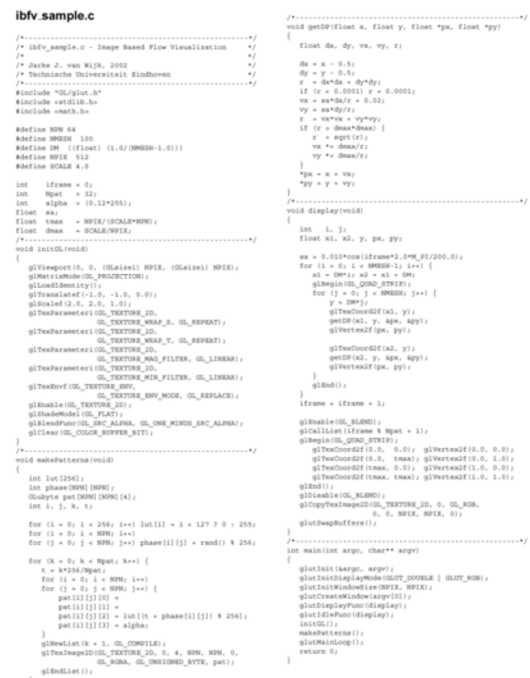
- what image?
  - random noise (gives standard LIC)

What Image?

Issues to consider

- aliasing in time and space
  - pink, not white noise
- contrast, boundaries





ibfv_sample.c

```
/*-----------------------------------------------------------------*/
/* ibfv_sample.c - Image Based Flow Visualization                  */
/*                                                                 */
/* Jarke J. van Wijk, 2002                                         */
/* Technische Universiteit Eindhoven                               */
/*-----------------------------------------------------------------*/
#include "GL/glut.h"
#include <stdlib.h>
#include <math.h>

#define NPN 64
#define NMESH  100
#define DM  ((float) (1.0/(NMESH-1.0)))
#define NPIX  512
#define SCALE 4.0

int     iframe = 0;
int     Npat   = 32;
int     alpha  = (0.12*255);
float   sa;
float   tmax   = NPIX/(SCALE*NPN);
float   dmax   = SCALE/NPIX;
/*-----------------------------------------------------------------*/
void initGL(void)
{
   glViewport(0, 0, (GLsizei) NPIX, (GLsizei) NPIX);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glTranslatef(-1.0, -1.0, 0.0);
   glScalef(2.0, 2.0, 1.0);
   glTexParameteri(GL_TEXTURE_2D,
                   GL_TEXTURE_WRAP_S, GL_REPEAT);
   glTexParameteri(GL_TEXTURE_2D,
                   GL_TEXTURE_WRAP_T, GL_REPEAT);
   glTexParameteri(GL_TEXTURE_2D,
                   GL_TEXTURE_MAG_FILTER, GL_LINEAR);
   glTexParameteri(GL_TEXTURE_2D,
                   GL_TEXTURE_MIN_FILTER, GL_LINEAR);
   glTexEnvf(GL_TEXTURE_ENV,
                   GL_TEXTURE_ENV_MODE, GL_REPLACE);
   glEnable(GL_TEXTURE_2D);
   glShadeModel(GL_FLAT);
   glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
   glClear(GL_COLOR_BUFFER_BIT);
}
/*-----------------------------------------------------------------*/
void makePatterns(void)
{
   int lut[256];
   int phase[NPN][NPN];
   GLubyte pat[NPN][NPN][4];
   int i, j, k, t;

   for (i = 0; i < 256; i++) lut[i] = i < 127 ? 0 : 255;
   for (i = 0; i < NPN; i++)
   for (j = 0; j < NPN; j++) phase[i][j] = rand() % 256;

   for (k = 0; k < Npat; k++) {
      t = k*256/Npat;
      for (i = 0; i < NPN; i++)
      for (j = 0; j < NPN; j++) {
          pat[i][j][0] =
          pat[i][j][1] =
          pat[i][j][2] = lut[(t + phase[i][j]) % 256];
          pat[i][j][3] = alpha;
      }
      glNewList(k + 1, GL_COMPILE);
      glTexImage2D(GL_TEXTURE_2D, 0, 4, NPN, NPN, 0,
                   GL_RGBA, GL_UNSIGNED_BYTE, pat);
      glEndList();
   }
}
/*-----------------------------------------------------------------*/
void getDP(float x, float y, float *px, float *py)
{
   float dx, dy, vx, vy, r;

   dx = x - 0.5;
   dy = y - 0.5;
   r  = dx*dx + dy*dy;
   if (r < 0.0001) r = 0.0001;
   vx = sa*dx/r + 0.02;
   vy = sa*dy/r;
   r  = vx*vx + vy*vy;
   if (r > dmax*dmax) {
      r  = sqrt(r);
      vx *= dmax/r;
      vy *= dmax/r;
   }
   *px = x + vx;
   *py = y + vy;
}
/*-----------------------------------------------------------------*/
void display(void)
{
   int   i, j;
   float x1, x2, y, px, py;

   sa = 0.010*cos(iframe*2.0*M_PI/200.0);
   for (i = 0; i < NMESH-1; i++) {
      x1 = DM*i; x2 = x1 + DM;
      glBegin(GL_QUAD_STRIP);
      for (j = 0; j < NMESH; j++) {
          y = DM*j;
          glTexCoord2f(x1, y);
          getDP(x1, y, &px, &py);
          glVertex2f(px, py);

          glTexCoord2f(x2, y);
          getDP(x2, y, &px, &py);
          glVertex2f(px, py);
      }
      glEnd();
   }
   iframe = iframe + 1;

   glEnable(GL_BLEND);
   glCallList(iframe % Npat + 1);
   glBegin(GL_QUAD_STRIP);
      glTexCoord2f(0.0,  0.0);  glVertex2f(0.0, 0.0);
      glTexCoord2f(0.0,  tmax); glVertex2f(0.0, 1.0);
      glTexCoord2f(tmax, 0.0);  glVertex2f(1.0, 0.0);
      glTexCoord2f(tmax, tmax); glVertex2f(1.0, 1.0);
   glEnd();
   glDisable(GL_BLEND);
   glCopyTexImage2D(GL_TEXTURE_2D, 0, GL_RGB,
                    0, 0, NPIX, NPIX, 0);
   glutSwapBuffers();
}
/*-----------------------------------------------------------------*/
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(NPIX, NPIX);
   glutCreateWindow(argv[0]);
   glutDisplayFunc(display);
   glutIdleFunc(display);
   initGL();
   makePatterns();
   glutMainLoop();
   return 0;
}
```

Nice Example

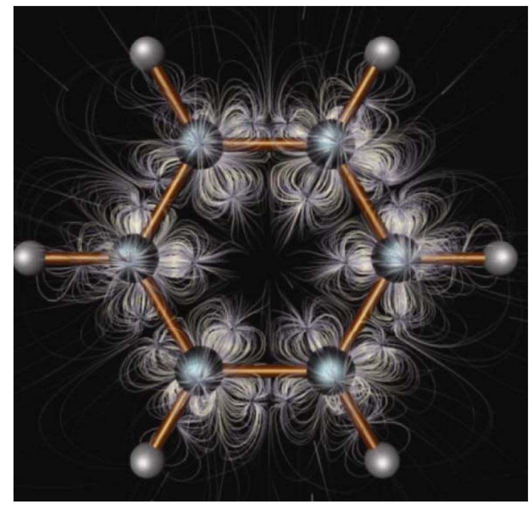
van Wijk's ibfv program

- testbed for visualization

Direct Visualization

Integral curves as shaded lines

- “hair” like analogy
- how to shade lines?
- usually objects co-D 1



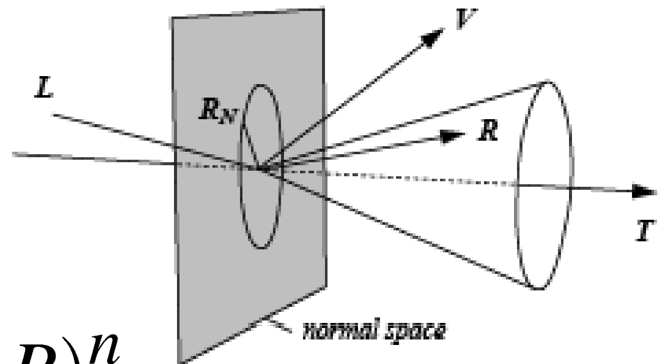
New shading model

- objects of co-D 2
- think cylinders of infinitesimal size

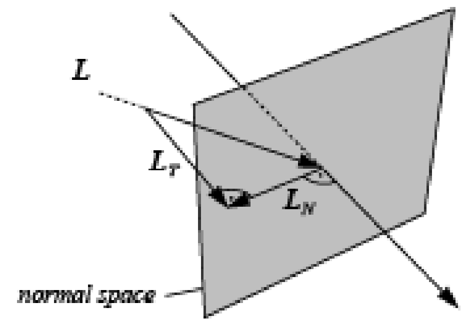
Shaded Lines

What normal?

- standard model



$$I = k_a + k_d(L \cdot N) + k_s(V \cdot R)^n$$



$$L \cdot N = |L_N| = \sqrt{1 - (L \cdot T)^2}$$

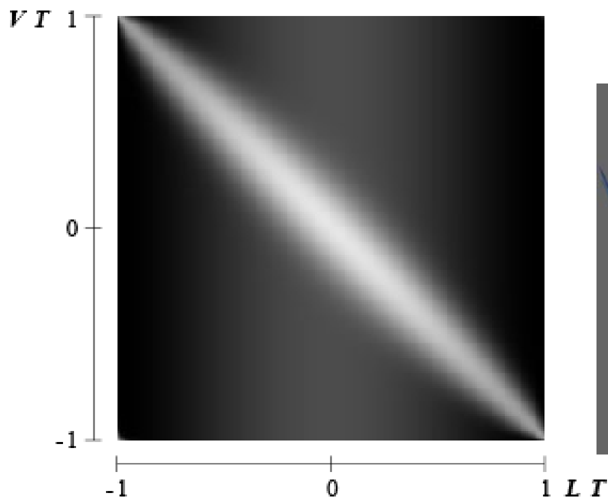
$$\begin{aligned} V \cdot R &= V \cdot (L_T - L_N) \\ &= \sqrt{1-(L \cdot T)^2}\sqrt{1-(V \cdot T)^2} - (L \cdot T)(V \cdot T) \end{aligned}$$

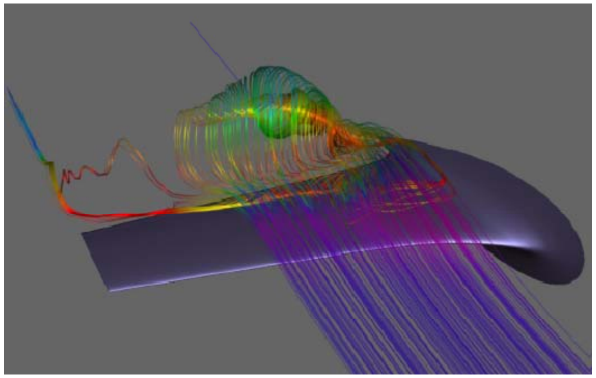
Implementation

Shading is function of L, T, V

- texture lookup
- inner products via texture xform

$$M = \frac{1}{2}\begin{pmatrix} L_1 & V_1 & 0 & 0 \\ L_2 & V_2 & 0 & 0 \\ L_3 & V_3 & 0 & 0 \\ 1 & 1 & 0 & 2 \end{pmatrix}$$





Details

Implementation

- shading too bright: for diffuse

$$I_d = k_d (L \cdot N)^p$$

- add transparency: “wispy” tails
- how to seed stream lines?
  - user driven
  - Monte Carlo; Voronoi
  - divergence... (reseed)

Papers

Image Based Flow Visualization

Jarke J. van Wijk*
Technische Universiteit Eindhoven
Dept. of Mathematics and Computer Science

Abstract

1 Introduction

2 Related work

Fast and Resolution Independent Line Integral Convolution

Detlev Stalling Hans-Christian Hege

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)[1]

Abstract

1 Introduction

249

Interactive Visualization Of 3D-Vector Fields Using Illuminated Stream Lines

Malte Zöckler, Detlev Stalling, Hans-Christian Hege

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)[1]

Abstract

1 Introduction