Motion estimation

Digital Visual Effects, Spring 2005

Yung-Yu Chuang

2005/3/23

with slides by Michael Black and P. Anandan

Announcements

DigiVFX

- Project #1 is due on next Tuesday, submission mechanism will be announced later this week.
- grading: report is important, results (good/bad), discussions on implementation, interface, features, etc.

Outline

DigiVFX

- Motion estimation
- Lucas-Kanade algorithm
- Tracking
- Optical flow

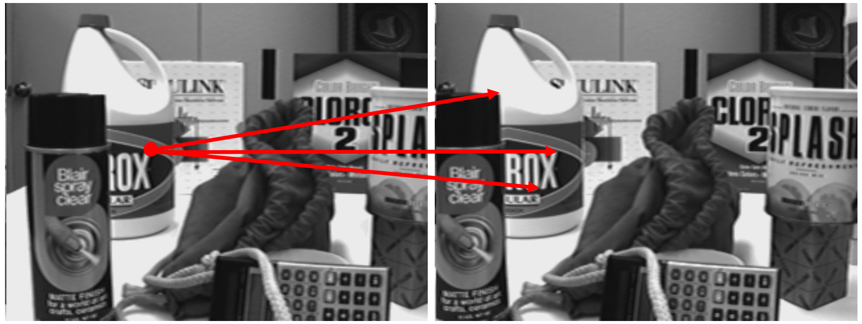
Motion estimation

DigiVFX

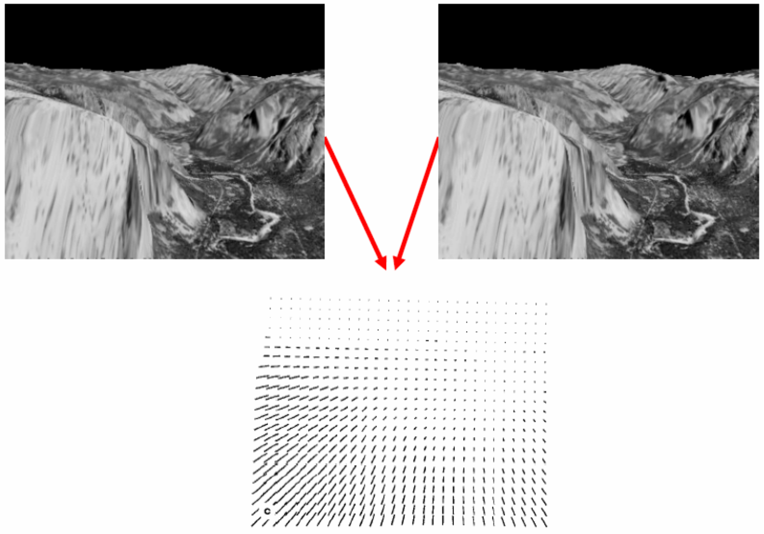
- Parametric motion (image alignment)
- Tracking
- Optical flow

Parametric motion

Tracking



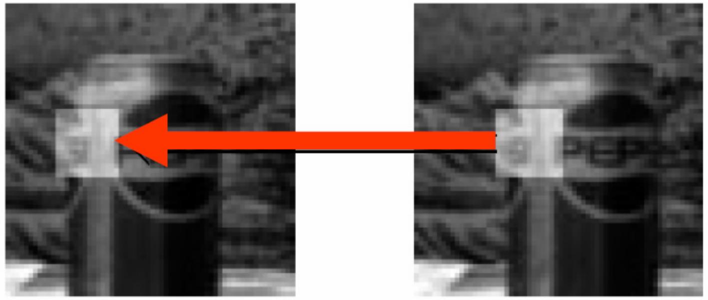
Optical flow



Three assumptions

- Brightness consistency
- Spatial coherence
- Temporal persistence

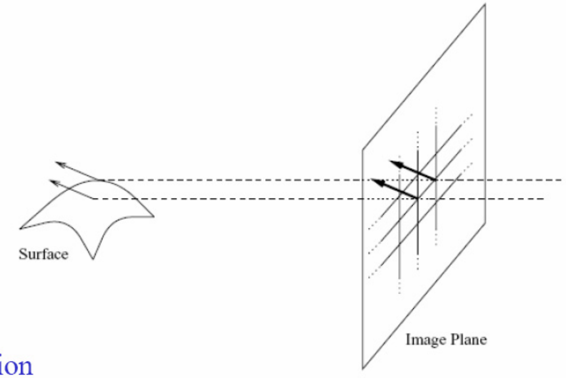
Brightness consistency



Assumption

Image measurements (e.g. brightness) in a small region remain the same although their location may change.

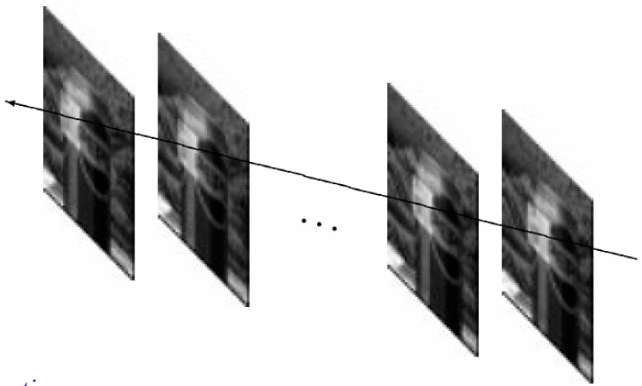
Spatial coherence



Assumption

* Neighboring points in the scene typically belong to the same surface and hence typically have similar motions.
* Since they also project to nearby points in the image, we expect spatial coherence in image flow.

Temporal persistence



Assumption:

The image motion of a surface patch changes gradually over time.

Image registration

Goal: register a template image $J(x)$ and an input image $I(x)$, where $x=(x,y)^T$.

Image alignment: $I(x)$ and $J(x)$ are two images

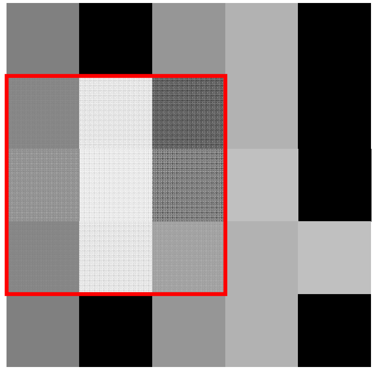
Tracking: $I(x)$ is the image at time t. $J(x)$ is a small patch around the point p in the image at $t+1$.

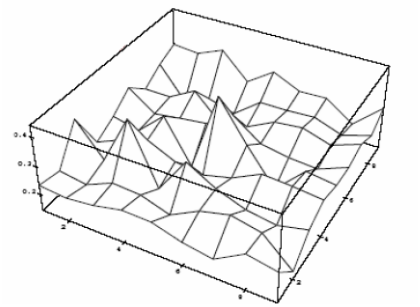
Optical flow: $I(x)$ and $J(x)$ are images of t and $t+1$.

Simple approach

- Minimize brightness difference

$$E(u,v)=\sum_{x,y}\left(I(x+u,y+v)-J(x,y)\right)^2$$





Simple SSD algorithm

For each offset *(u, v)*

compute *E(u,v);*

Choose *(u, v)* which minimizes *E(u,v);*

Problems:

- Not efficient
- No sub-pixel accuracy

Lucas-Kanade algorithm

Newton's method

- Root finding for $f(x)=0$

Taylor's expansion:

$$f(x_0+\epsilon)=f(x_0)+f'(x_0)\epsilon+\tfrac{1}{2}f''(x_0)\epsilon^2+\ldots.$$

$$f(x_0+\epsilon)\approx f(x_0)+f'(x_0)\epsilon.$$

$$\epsilon_n=-\frac{f(x_n)}{f'(x_n)}.$$

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$$

Lucas-Kanade algorithm

$$E(u,v)=\sum_{x,y}\left(I(x+u,y+v)-J(x,y)\right)^2$$

$$\boxed{I(x+u,y+v)\approx I(x,y)+uI_x+vI_y}$$

$$=\sum_{x,y}\left(I(x,y)-J(x,y)+uI_x+vI_y\right)^2$$

$$0=\frac{\partial E}{\partial u}=\sum_{x,y}2I_x\left(I(x,y)-J(x,y)+uI_x+vI_y\right)$$

$$0=\frac{\partial E}{\partial v}=\sum_{x,y}2I_y\left(I(x,y)-J(x,y)+uI_x+vI_y\right)$$

Lucas-Kanade algorithm

$$0=\frac{\partial E}{\partial u}=\sum_{x,y}2I_x\left(I(x,y)-J(x,y)+uI_x+vI_y\right)$$

$$0=\frac{\partial E}{\partial v}=\sum_{x,y}2I_y\left(I(x,y)-J(x,y)+uI_x+vI_y\right)$$

$$\Rightarrow \begin{cases}\sum_{x,y}I_x^2u+I_xI_yv=\sum_{x,y}I_x\left(J(x,y)-I(x,y)\right)\\ \sum_{x,y}I_xI_yu+I_y^2v=\sum_{x,y}I_y\left(J(x,y)-I(x,y)\right)\end{cases}$$

$$\Rightarrow \begin{bmatrix}\sum_{x,y}I_x^2 & \sum_{x,y}I_xI_y\\ \sum_{x,y}I_xI_y & \sum_{x,y}I_y^2\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}=\begin{bmatrix}\sum_{x,y}I_x\left(J(x,y)-I(x,y)\right)\\ \sum_{x,y}I_y\left(J(x,y)-I(x,y)\right)\end{bmatrix}$$

Lucas-Kanade algorithm

iterate
- shift I(x,y) with (u,v)
- compute gradient image I_x, I_y
- compute error image J(x,y)-I(x,y)
- compute Hessian matrix
- solve the linear system
- (u,v)=(u,v)+(Δu,Δv)

until converge

$$\begin{bmatrix}\sum_{x,y}I_x^2 & \sum_{x,y}I_xI_y\\ \sum_{x,y}I_xI_y & \sum_{x,y}I_y^2\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}=\begin{bmatrix}\sum_{x,y}I_x\left(J(x,y)-I(x,y)\right)\\ \sum_{x,y}I_y\left(J(x,y)-I(x,y)\right)\end{bmatrix}$$

Parametric model

$$E(u,v)=\sum_{x,y}\left(I(x+u,y+v)-J(x,y)\right)^2$$

$$\Rightarrow E(\mathbf{p})=\sum_{\mathbf{x}}\left(I(\mathbf{W}(\mathbf{x};\mathbf{p}))-J(\mathbf{x})\right)^2$$

translation $\quad \mathbf{W}(\mathbf{x};\mathbf{p})=\begin{pmatrix}x+d_x\\ y+d_y\end{pmatrix},\ p=(d_x,d_y)^T$

affine $\quad \mathbf{W}(\mathbf{x};\mathbf{p})=\mathbf{Ax}+\mathbf{d}=\begin{pmatrix}1+d_{xx} & d_{xy} & d_x\\ d_{yx} & 1+d_{yy} & d_y\end{pmatrix}\begin{pmatrix}x\\ y\\ 1\end{pmatrix},$

$$p=(d_{xx},d_{xy},d_{yx},d_{yy},d_x,d_y)^T$$

Parametric model

DigiVFX

minimize $\sum_{\mathbf{x}} \left(I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - J(\mathbf{x})\right)^2$

with respect to $\Delta\mathbf{p}$

$$\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p}) \approx \mathbf{W}(\mathbf{x};\mathbf{p}) + \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\Delta\mathbf{p}$$

$$I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) \approx I(\mathbf{W}(\mathbf{x};\mathbf{p}) + \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\Delta\mathbf{p})$$

$$\approx I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \frac{\partial I}{\partial \mathbf{x}}\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\Delta\mathbf{p}$$

$\Rightarrow$ minimize $\sum_{\mathbf{x}} \left(I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\Delta\mathbf{p} - J(\mathbf{x})\right)^2$

Parametric model

DigiVFX

warped image

image gradient

$$\sum_{\mathbf{x}} \left(\boxed{I(\mathbf{W}(\mathbf{x};\mathbf{p}))} + \boxed{\nabla I}\boxed{\frac{\partial \mathbf{W}}{\partial \mathbf{p}}}\Delta\mathbf{p} - J(\mathbf{x})\right)^2$$

Jacobian of the warp

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}} \\ \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_n} \\ \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_n} \end{pmatrix}$$

Jacobian of the warp

DigiVFX

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}} \\ \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{W}_x}{\partial \mathbf{p}_n} \\ \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{W}_y}{\partial \mathbf{p}_n} \end{pmatrix}$$

For example, for affine

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} 1+d_{xx} & d_{xy} & d_x \\ d_{yx} & 1+d_{yy} & d_y \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} (1+d_{xx})x + d_{xy}y + d_x \\ d_{yx}x + (1+d_{yy})y + d_y \end{pmatrix}$$

$\Rightarrow$ $\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}$

Parametric model

DigiVFX

minimize $\sum_{\mathbf{x}} \left(I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\Delta\mathbf{p} - J(\mathbf{x})\right)^2$

$\Rightarrow$ $0 = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\Delta\mathbf{p} - J(\mathbf{x})\right]$

$$\Delta\mathbf{p} = \mathbf{H}^{-1}\sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[J(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right]$$

Hessian $\mathbf{H} = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$

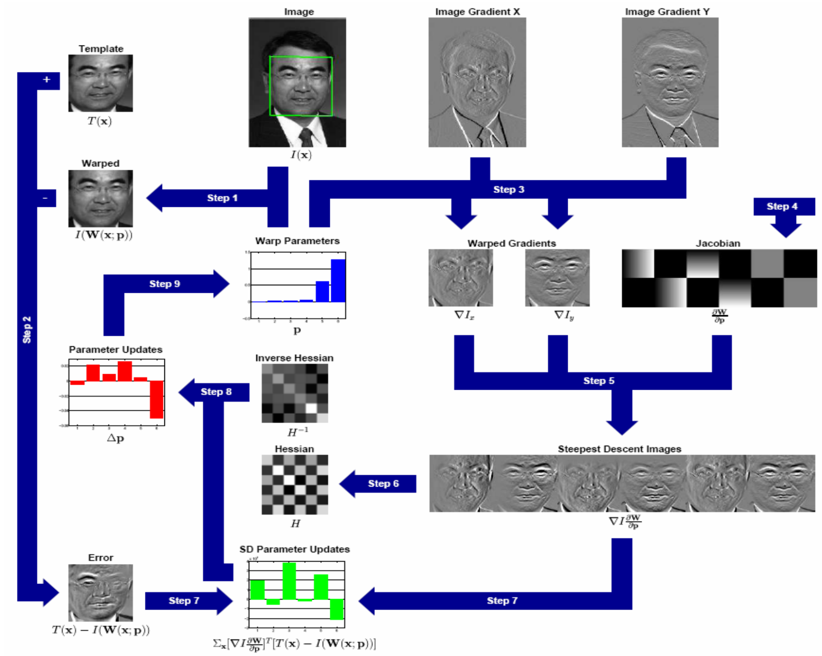
Lucas-Kanade algorithm

iterate

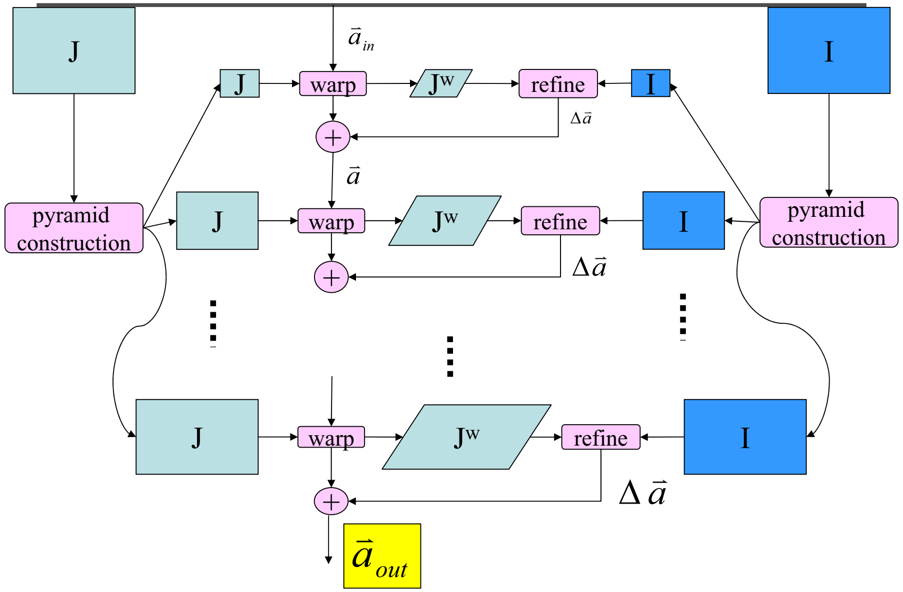
- warp I with W(x;p)
- compute error image J(x,y)-I(W(x,p))
- compute gradient image
- evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at (x;p)
- compute $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
- compute Hessian
- compute $\sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [J(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$
- solve $\Delta \mathbf{p}$
- update p by p+ $\Delta \mathbf{p}$

until converge

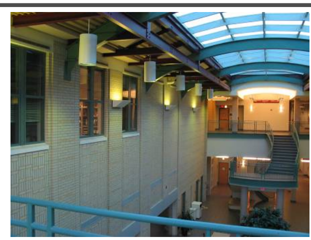
$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [J(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$



Coarse-to-fine strategy



Application of image alignment



Tracking

Tracking


Tracking

brightness constancy $I(x+u, y+v, t+1) - I(x, y, t) = 0$

$$I(x,y,t) + uI_x(x,y,t) + vI_y(x,y,t) + I_t(x,y,t) - I(x,y,t) \approx 0$$

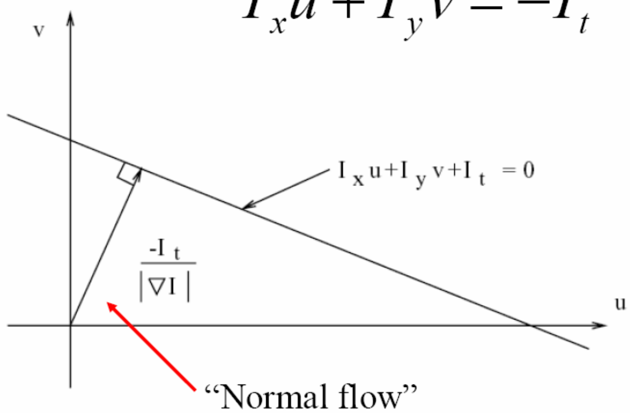
$$uI_x(x,y,t) + vI_y(x,y,t) + I_t(x,y,t) = 0$$

$I_x u + I_y v + I_t = 0$ optical flow constraint equation

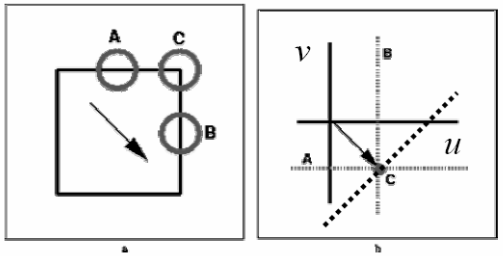
Optical flow constraint equation

At a single image pixel, we get a line:

$$I_x u + I_y v = -I_t$$



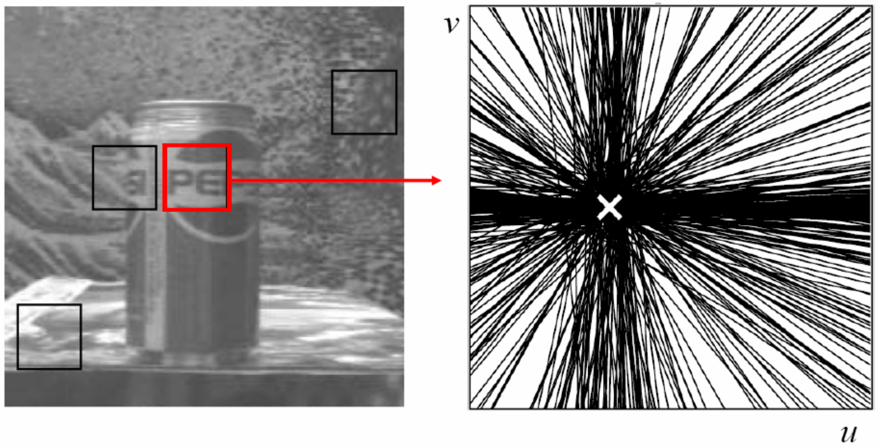
Multiple constraint



Combine constraints to get an estimate of velocity.

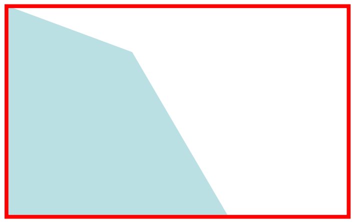
Area-based method

- Assume spatial smoothness

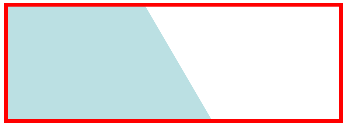


Aperture problem

Aperture problem



Aperture problem



Demo for aperture problem

- http://www.sandlotscience.com/Distortions/Breathing_objects.htm
- http://www.sandlotscience.com/Ambiguous/barberpole.htm

Aperture problem

- Larger window reduces ambiguity, but easily violates spatial smoothness assumption

Area-based method

- Assume spatial smoothness

$$E(u,v)=\sum_{x,y}\left(I_x u+I_y v+I_t\right)^2$$

$$\frac{\partial E}{\partial u}=\sum_R (I_x u+I_y v+I_t)I_x=0$$

$$\frac{\partial E}{\partial v}=\sum_R (I_x u+I_y v+I_t)I_y=0$$

Area-based method

$$\left[\sum_R I_x^2\right]u + \left[\sum_R I_x I_y\right]v = -\sum_R I_x I_t$$

$$\left[\sum_R I_x I_y\right]u + \left[\sum_R I_y^2\right]v = -\sum_R I_y I_t$$

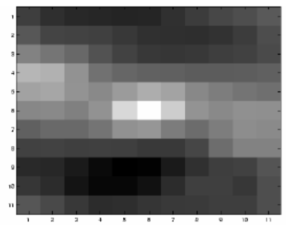
$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

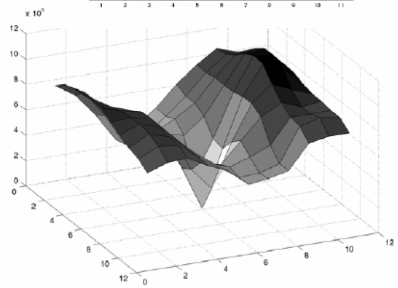
must be invertible

Area-based method

- The eigenvalues tell us about the local image structure.
- They also tell us how well we can estimate the flow in both directions
- Link to Harris corner detector

Textured area

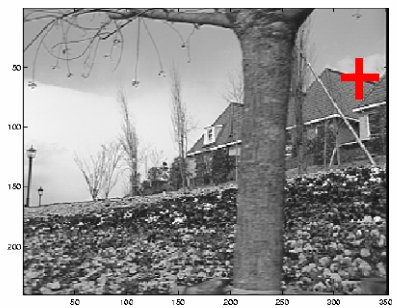


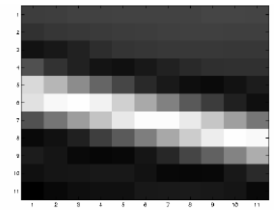


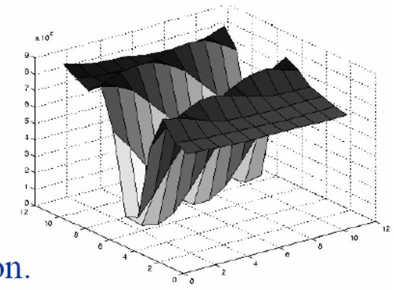
$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}$$

Gradients in x and y.

Edge



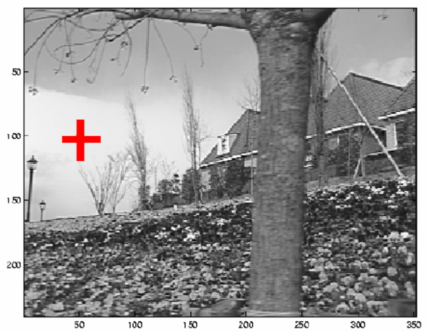


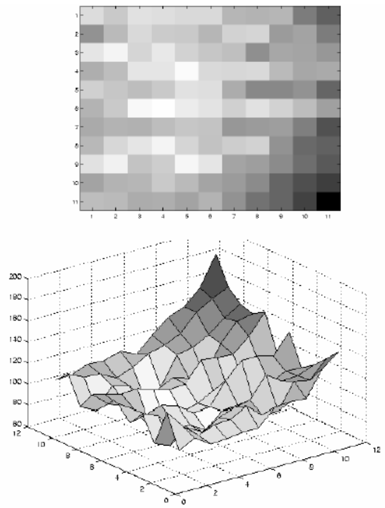


$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}$$

Gradients oriented in one direction.

Homogenous area



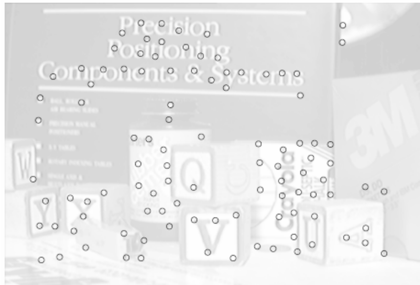


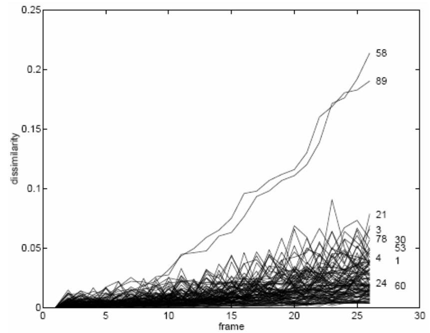
$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}$$

Weak gradients everywhere.

KLT tracking

- Select feature by $min(\lambda_1, \lambda_2) > \lambda$
- Monitor features by measuring dissimilarity





Translational Model

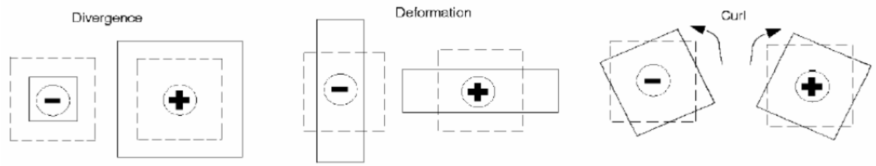
What's wrong with the translational assumption (ie constant motion within a region R)?

How can we generalize it?

Affine Flow

$$E(\mathbf{a}) = \sum_{x,y \in R} (\nabla I^T \mathbf{u}(\mathbf{x};\mathbf{a}) + I_t)^2$$

$$\mathbf{u}(\mathbf{x};\ \mathbf{a}) = \begin{bmatrix} u(\mathbf{x};\mathbf{a}) \\ v(\mathbf{x};\mathbf{a}) \end{bmatrix} = \begin{bmatrix} a_1 + a_2 x + a_3 y \\ a_4 + a_5 x + a_6 y \end{bmatrix}$$



Optimization

$$E(\mathbf{a}) = \sum_{x,y \in R} (I_x a_1 + I_x a_2 x + I_x a_3 y + I_y a_4 + I_y a_5 x + I_y a_6 y + I_t)^2$$

Differentiate wrt the a_i and set equal to zero.

$$\begin{bmatrix} \Sigma I_x^2 & \Sigma I_x^2 x & \Sigma I_x^2 y & \Sigma I_x I_y & \Sigma I_x I_y x & \Sigma I_x I_y y \\ \Sigma I_x^2 x & \Sigma I_x^2 x^2 & \Sigma I_x^2 xy & \Sigma I_x I_y x & \Sigma I_x I_y x^2 & \Sigma I_x I_y xy \\ & & & \vdots & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} -\Sigma I_x I_t \\ -\Sigma I_x I_t x \\ -\Sigma I_x I_t y \\ -\Sigma I_y I_t \\ -\Sigma I_y I_t x \\ -\Sigma I_y I_t y \end{bmatrix}$$

KLT tracking

http://www.ces.clemson.edu/~stb/klt/

KLT tracking

http://www.ces.clemson.edu/~stb/klt/

SIFT tracking (matching actually)

Frame 0 → Frame 10

SIFT tracking

Frame 0 → Frame 100

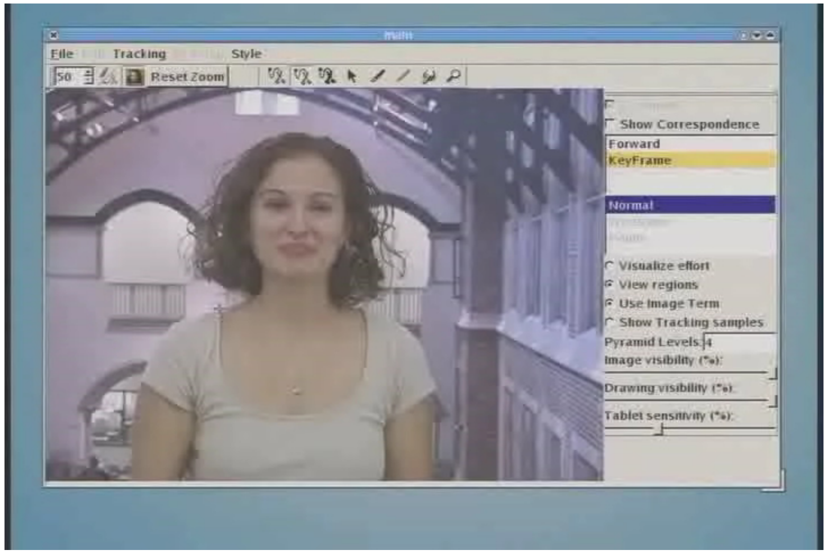
SIFT tracking

Frame 0 → Frame 200

KLT vs SIFT tracking

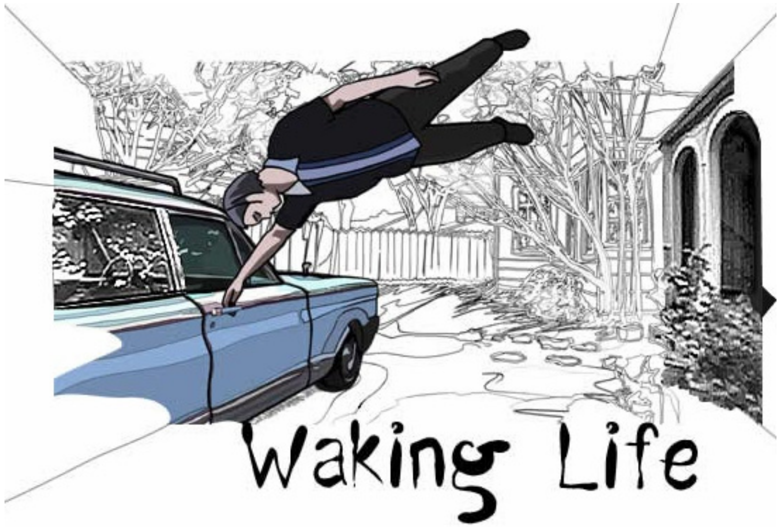
- KLT has larger accumulating error; partly because our KLT implementation doesn't have affine transformation?
- SIFT is surprisingly robust

Tracking for rotoscoping



Tracking for rotoscoping

Waking life



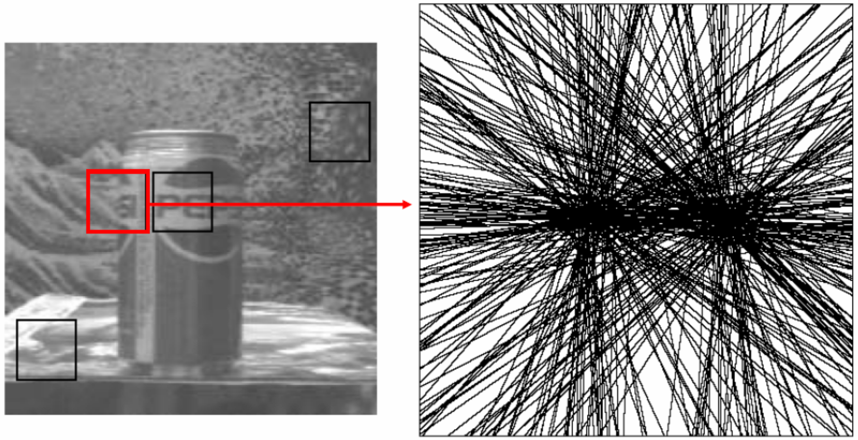
Optical flow

Single-motion assumption

Violated by

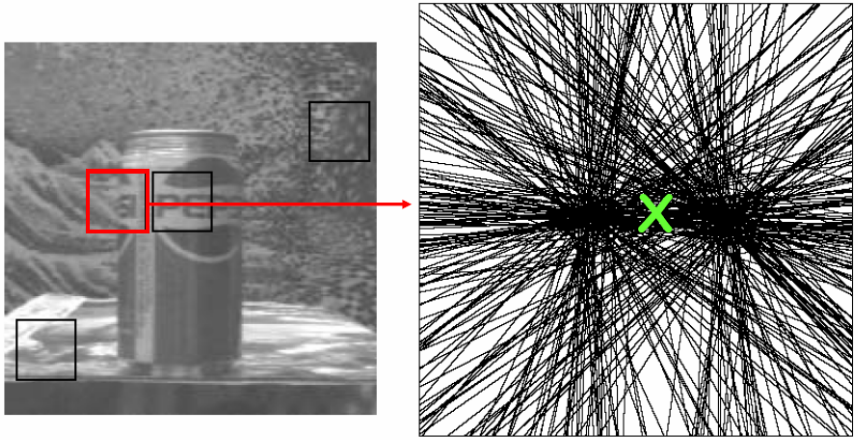
- Motion discontinuity
- Shadows
- Transparency
- Specular reflection
- …

Multiple motion



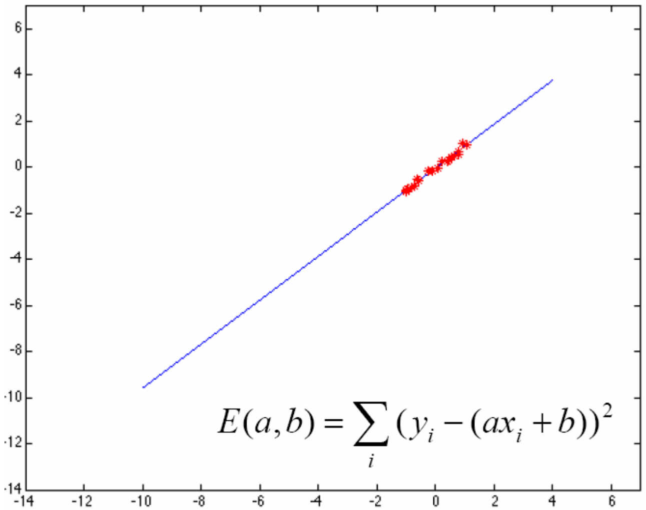
What is the “best” fitting translational motion?

Multiple motion



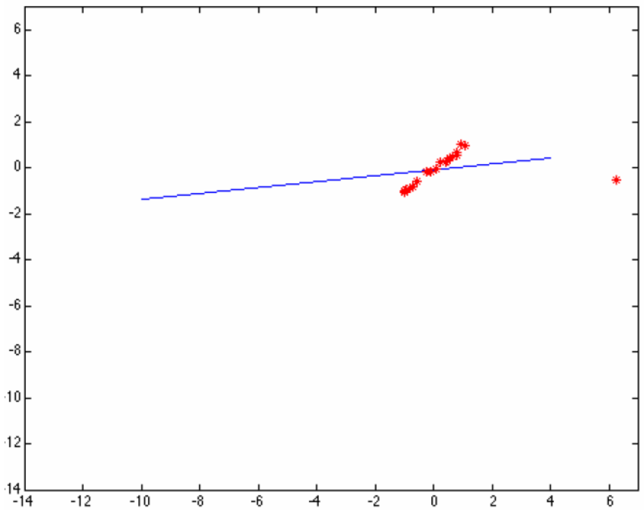
Least squares fit.

Simple problem: fit a line

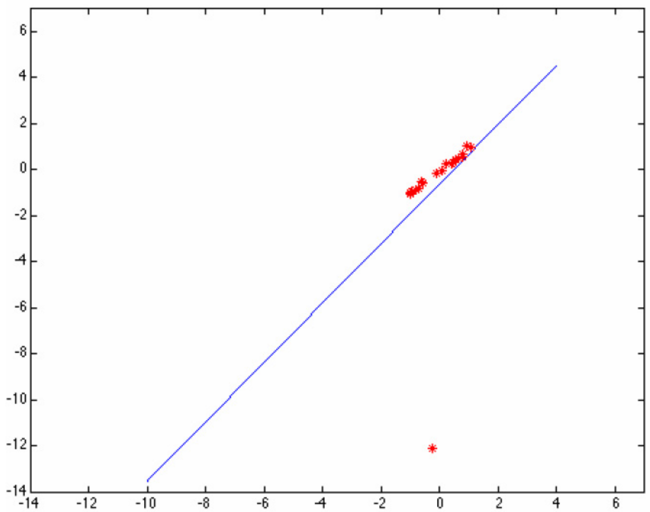


$$E(a,b)=\sum_i (y_i-(ax_i+b))^2$$

Least-square fit



Least-square fit

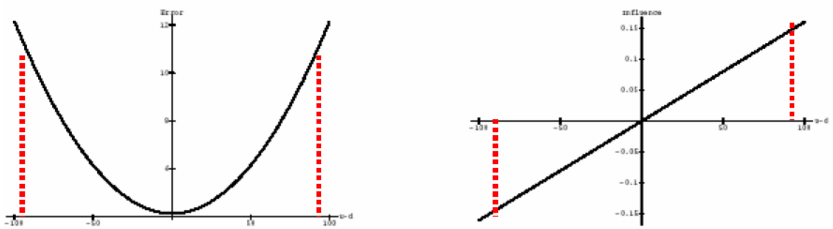


Robust statistics

- Recover the best fit for the majority of the data
- Detect and reject outliers

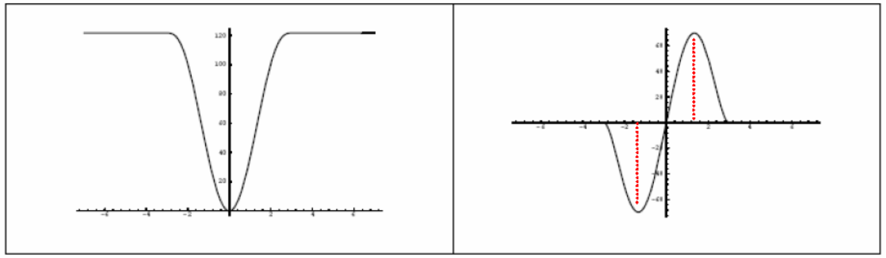
Approach

Influence is proportional to the derivative of the ρ function.



Want to give less influence to points beyond some value.

Robust weighting



Tukey's biweight.

Beyond a point, the influence begins to decrease.

Beyond where the second derivative is zero – outlier points

Robust estimation

$$E(\mathbf{a}) = \sum_{x,y \in R} \rho(I_x u + I_y v + I_t, \sigma)$$

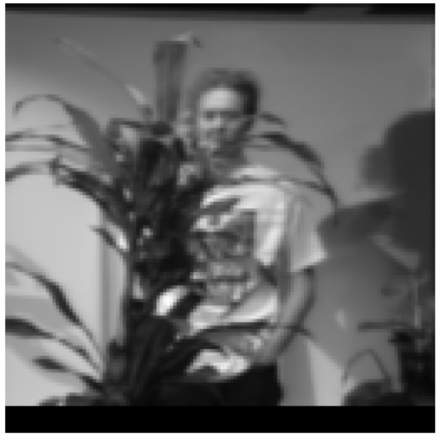
Minimize: differentiate and set equal to zero:

$$\frac{\partial E}{\partial u} = \sum_{x,y \in R} \psi(I_x u + I_y v + I_t, \sigma) I_x = 0$$

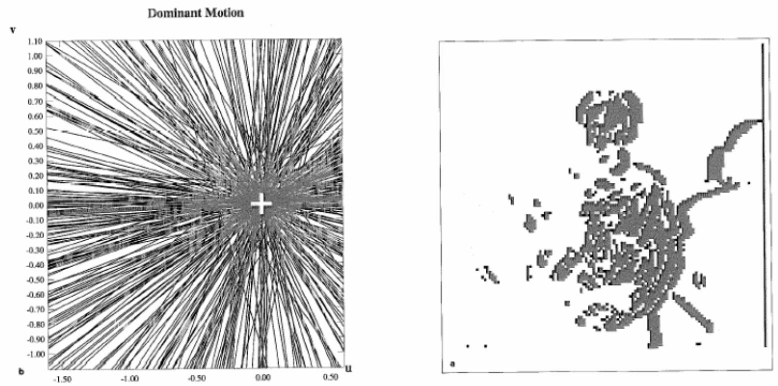
$$\frac{\partial E}{\partial v} = \sum_{x,y \in R} \psi(I_x u + I_y v + I_t, \sigma) I_y = 0$$

No closed form solution!

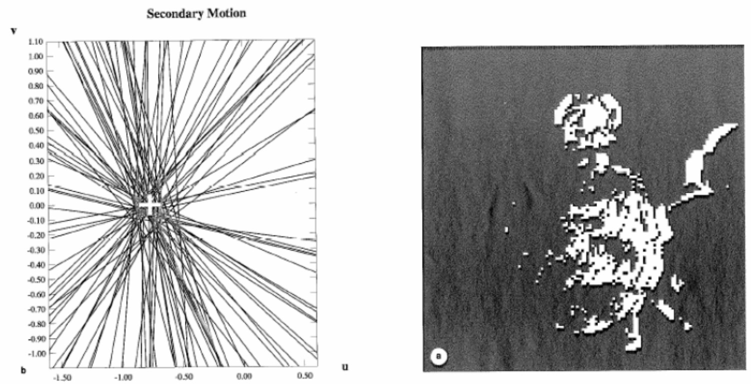
Fragmented Occlusion



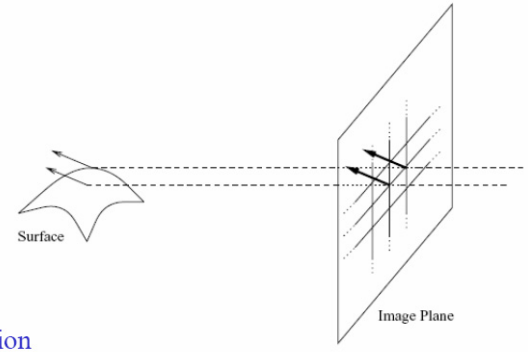
Results



Results



Regularization and dense optical flow

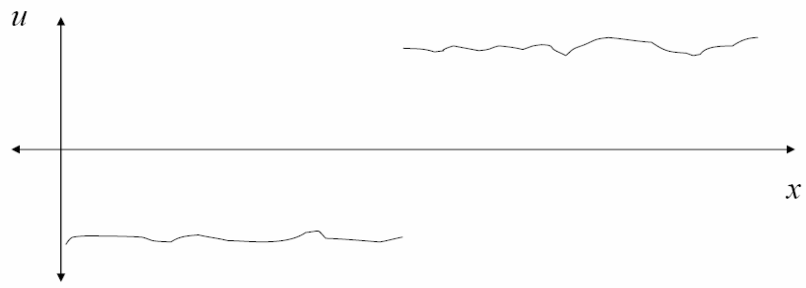


Assumption

- Neighboring points in the scene typically belong to the same surface and hence typically have similar motions.
- Since they also project to nearby points in the image, we expect spatial coherence in image flow.

Formalize this Idea

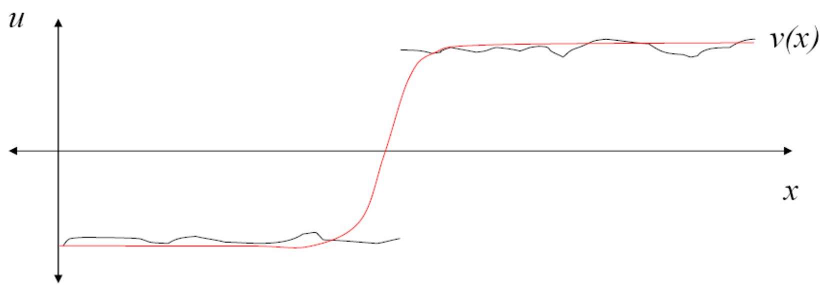
Noisy 1D signal:



Noisy measurements $u(x)$

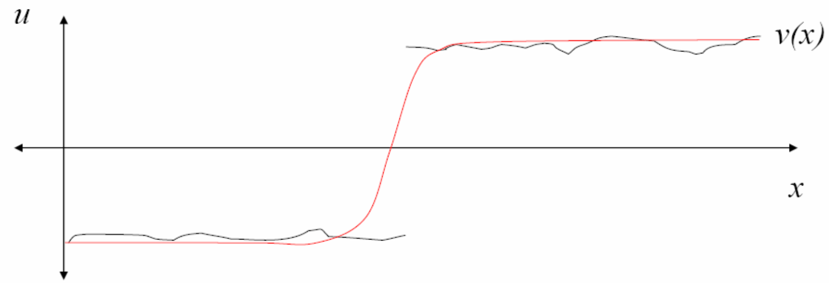
Regularization

Find the "best fitting" smoothed function $v(x)$



Noisy measurements $u(x)$

Regularization



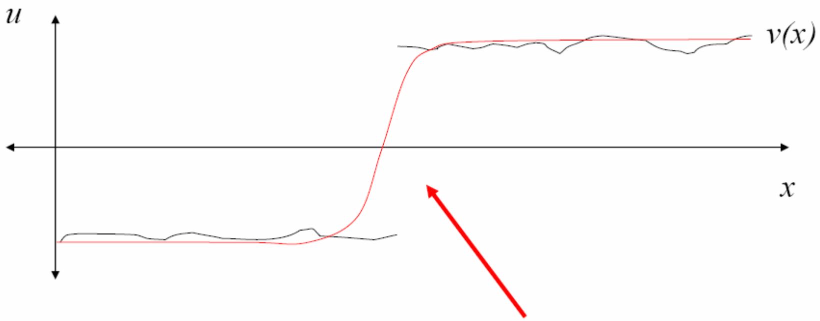
Minimize:

Faithful to the data

Spatial smoothness assumption

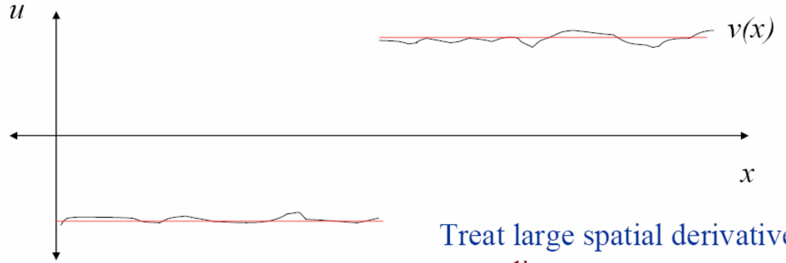
$$E(v)=\sum_{x=1}^{N}(v(x)-u(x))^2+\lambda\sum_{x=1}^{N-1}(v(x+1)-v(x))^2$$

Discontinuities



What about this discontinuity?
What is happening here?
What can we do?

Robust Regularization

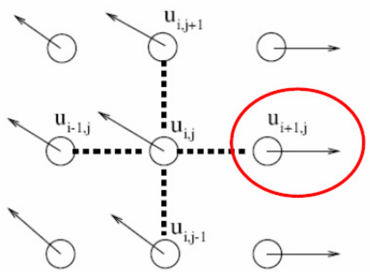


Treat large spatial derivatives as outliers.

Minimize:

$$E(v)=\sum_{x=1}^{N}\rho(v(x)-u(x),\sigma_1)+\lambda\sum_{x=1}^{N-1}\rho(v(x+1)-v(x),\sigma_2)$$

Optical flow



Outlier with respect to neighbors.

Robust formulation of spatial coherence term

$$E_S(u,v)=\rho(u_x)+\rho(u_y)+\rho(v_x)+\rho(v_y)$$

"Dense" Optical Flow

$$E_D(\mathbf{u}(\mathbf{x}))=\rho(I_x(\mathbf{x})u(\mathbf{x})+I_y(\mathbf{x})v(\mathbf{x})+I_t(\mathbf{x}),\sigma_D)$$

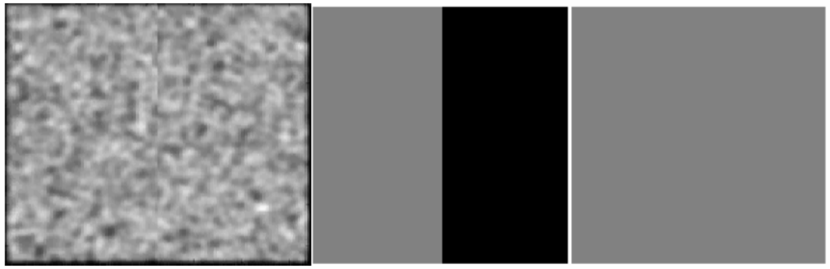
$$E_S(u,v)=\sum_{\mathbf{y}\in G(\mathbf{x})}[\rho(u(\mathbf{x})-u(\mathbf{y}),\sigma_S)+\rho(v(\mathbf{x})-v(\mathbf{y}),\sigma_S)]$$

Objective function:

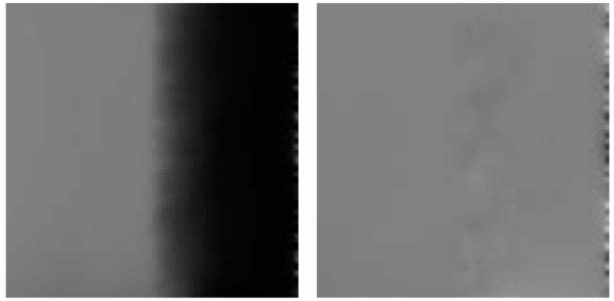
$$E(\mathbf{u})=\sum_{\mathbf{x}}E_D(\mathbf{u}(\mathbf{x}))+\lambda E_S(\mathbf{u}(\mathbf{x}))$$

When ρ is quadratic = "Horn and Schunck"

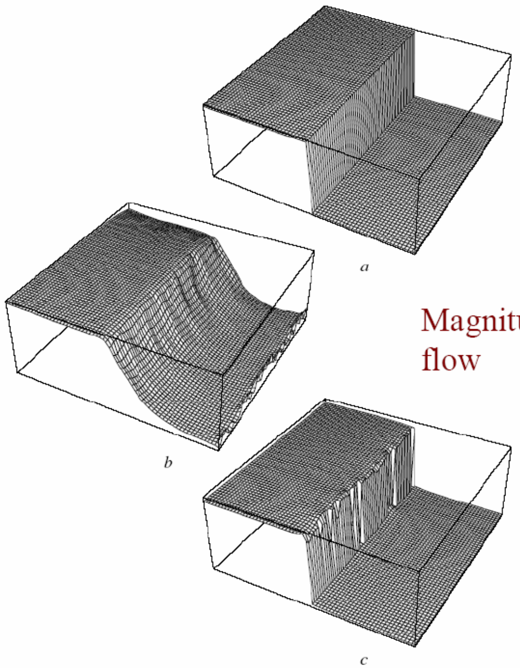
Example



Quadratic:

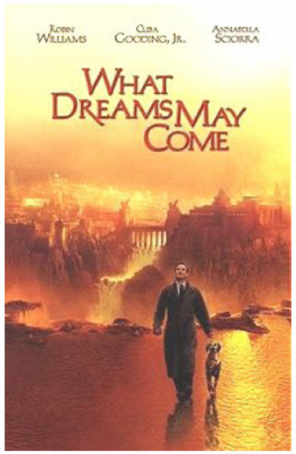


Robust:



Magnitude of horizontal flow

Applications of Optical Flow

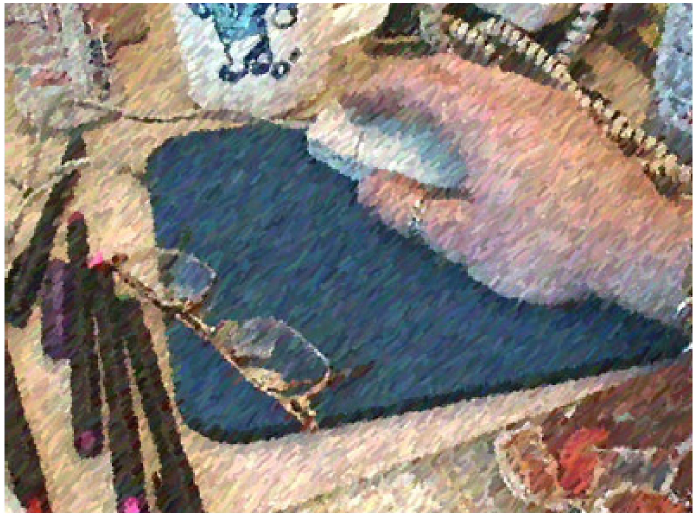


Impressionist effect.
Transfer motion of real world to a painting

Input for the NPR algorithm

Brushes

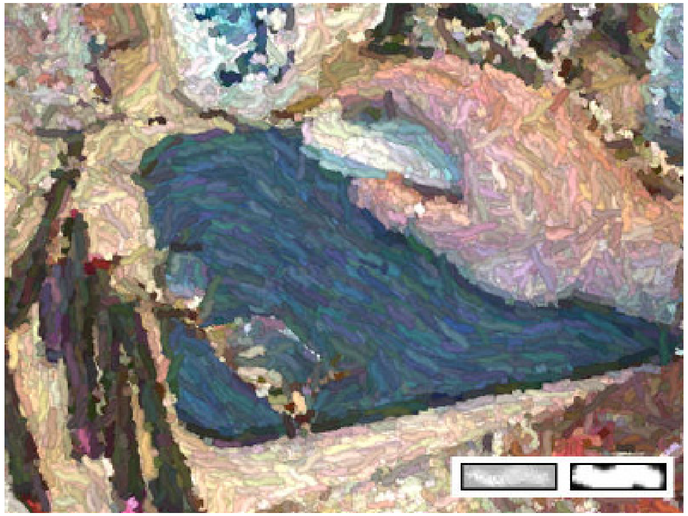
Edge clipping



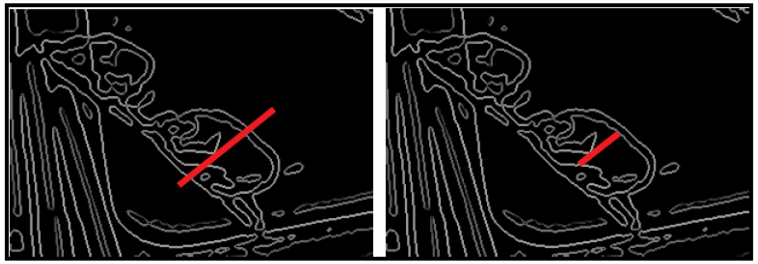
Gradient

Smooth gradient

Textured brush



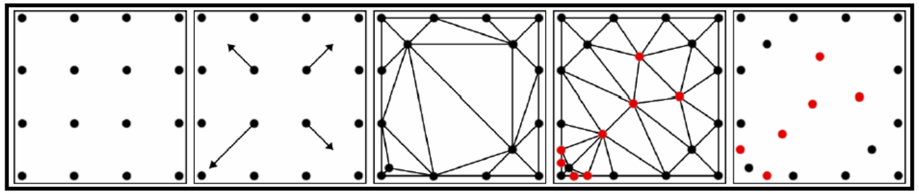
Edge clipping



Temporal artifacts

Frame-by-frame application of the NPR algorithm

Temporal coherence



RE:Vision

What dreams may come

Reference

- B.D. Lucas and T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision, Proceedings of the 1981 DARPA Image Understanding Workshop, 1981, pp121-130.
- Bergen, J. R. and Anandan, P. and Hanna, K. J. and Hingorani, R., Hierarchical Model-Based Motion Estimation, ECCV 1992, pp237-252.
- J. Shi and C. Tomasi, Good Features to Track, CVPR 1994, pp593-600.
- Michael Black and P. Anandan, The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields, Computer Vision and Image Understanding 1996, pp75-104.
- S. Baker and I. Matthews, Lucas-Kanade 20 Years On: A Unifying Framework, International Journal of Computer Vision, 56(3), 2004, pp221 - 255.
- Peter Litwinowicz, Processing Images and Video for An Impressionist Effects, SIGGRAPH 1997.
- Aseem Agarwala, Aaron Hertzman, David Salesin and Steven Seitz, Keyframe-Based Tracking for Rotoscoping and Animation, SIGGRAPH 2004, pp584-591.