# Motion estimation

Digital Visual Effects

*Yung-Yu Chuang*

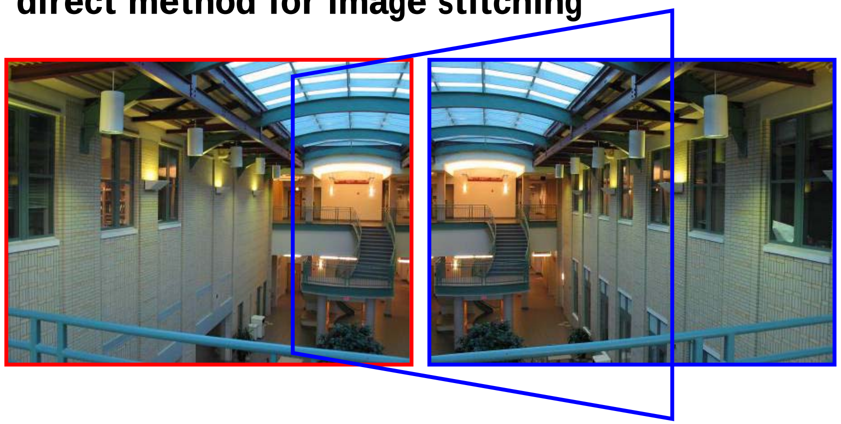
*with slides by Michael Black and P. Anandan*

## Motion estimation

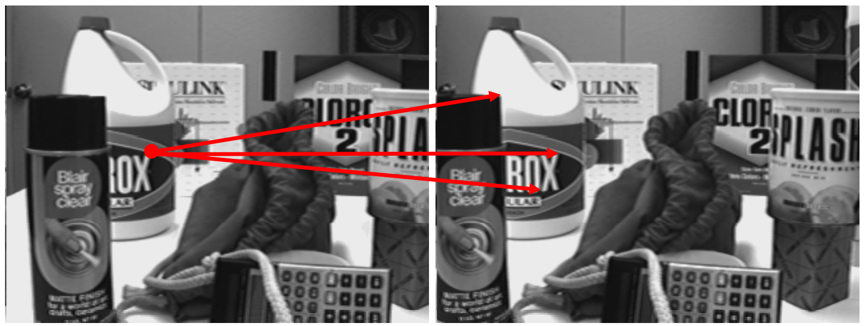
- Parametric motion (image alignment)
- Tracking
- Optical flow

## Parametric motion

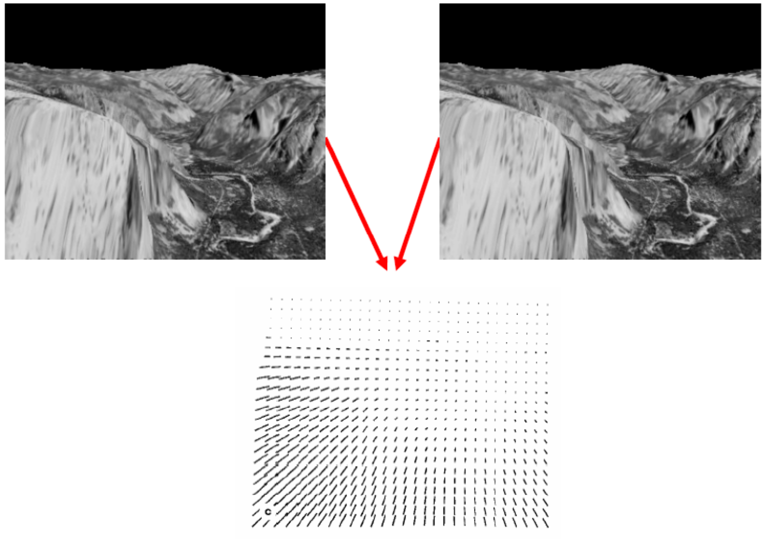
**direct method for image stitching**



## Tracking



## Optical flow



## Three assumptions

- Brightness consistency
- Spatial coherence
- Temporal persistence

## Brightness consistency

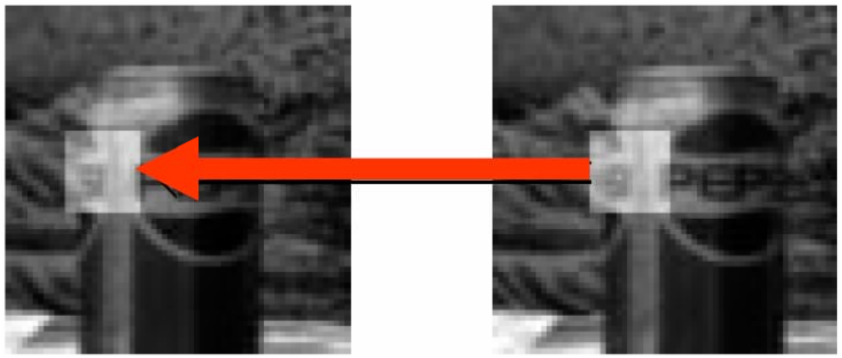
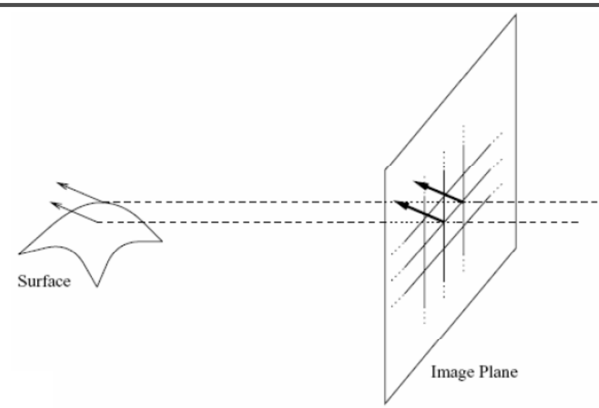


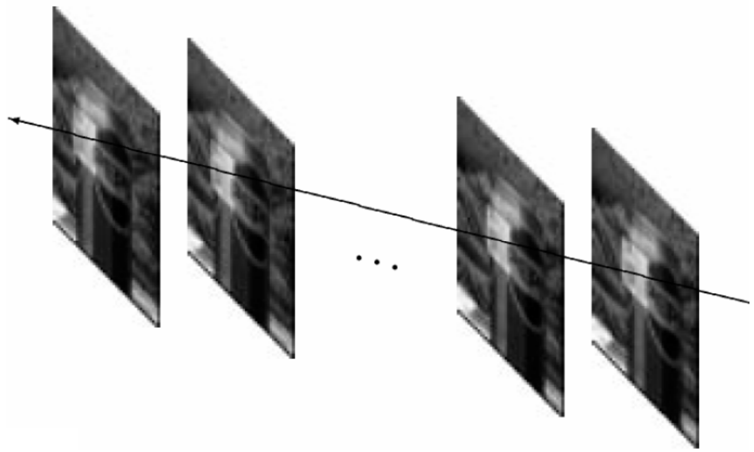
Image measurement (e.g. brightness) in a small region remain the same although their location may change.

## Spatial coherence




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

## Temporal persistence



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

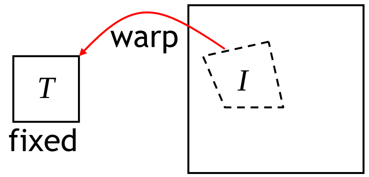
## Image registration

Goal: register a template image $T(x)$ and an input image $I(x)$, where $x=(x,y)^T$. (warp $I$ so that it matches $T$)

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

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

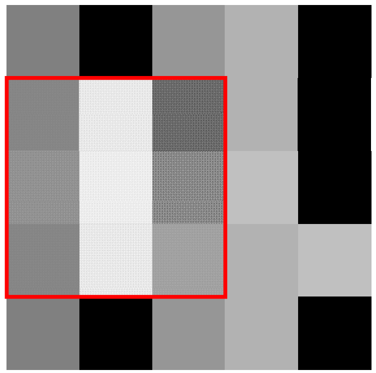
Optical flow: $T(x)$ and $I(x)$ are patches of images at $t$ and $t+1$.

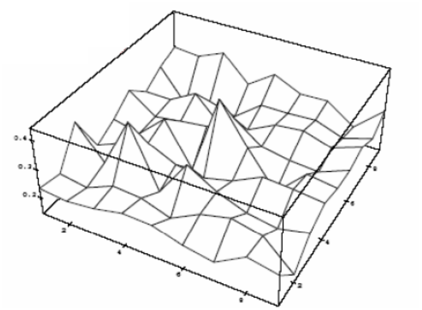



## Simple approach (for translation)

- Minimize brightness difference

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





## Simple SSD algorithm

For each offset $(u, v)$

compute $E(u,v)$;

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

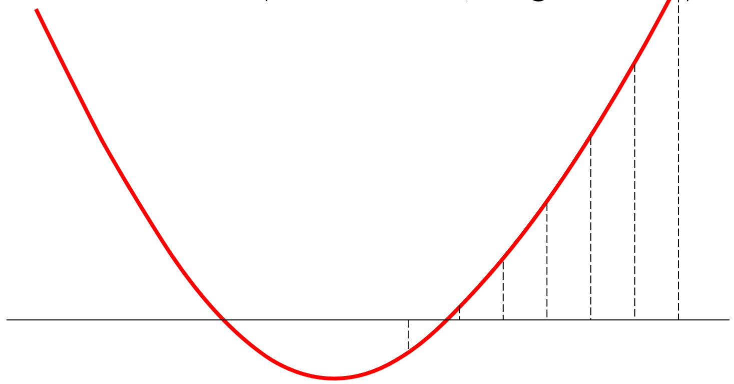
Problems:

- Not efficient
- No sub-pixel accuracy

# Lucas-Kanade algorithm

## Newton's method

- Root finding for *f(x)=0*
- March x and test signs
- Determine Δx (small→slow; large→ miss)



## Newton's method

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

## Newton's method

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

Taylor's expansion:

$$f(x_0+\varepsilon)=f(x_0)+f'(x_0)\varepsilon+\frac{1}{2}f''(x_0)\varepsilon^2+\ldots$$

$$f(x_0+\varepsilon)\approx f(x_0)+f'(x_0)\varepsilon$$

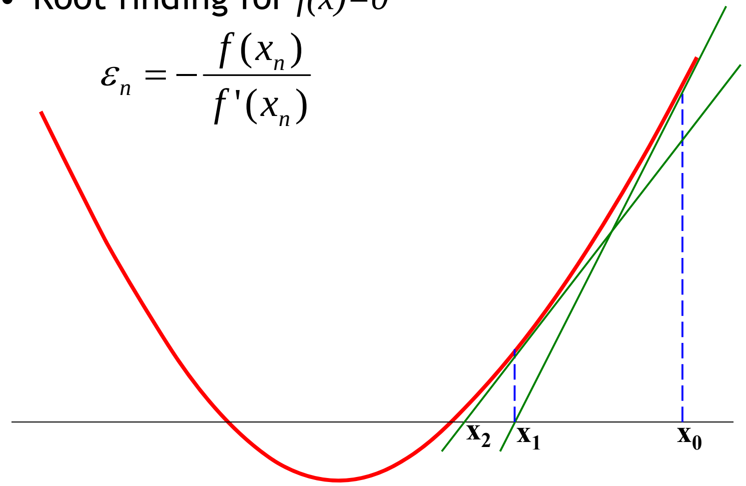
$$\varepsilon_n=-\frac{f(x_n)}{f'(x_n)}$$

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

## Newton's method

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

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




## Newton's method

pick up $\mathbf{x}=\mathbf{x}_0$

iterate

compute $\Delta x = -\frac{f(x)}{f'(x)}$

update $\mathbf{x}$ by $\mathbf{x}+\boldsymbol{\Delta}\mathbf{x}$

until converge

Finding root is useful for optimization because

Minimize $g(x) \rightarrow$ find root for $f(x)=g'(x)=0$

## Lucas-Kanade algorithm

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

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

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

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

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

## Lucas-Kanade algorithm

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

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

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

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

## Lucas-Kanade algorithm

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

until converge

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

## Parametric model

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

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

Our goal is to find $\mathbf{p}$ to minimize $\mathbf{E(p)}$

for all $\mathbf{x}$ in $T$'s domain

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

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

## Parametric model

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

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

## Parametric model

warped image

target image

image gradient

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

*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 matrix

- The Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function.

$$F(x_1, x_2, \dots x_n) \qquad F : \mathbf{R}^n \rightarrow \mathbf{R}^m$$

$$= (f_1(x_1, x_2, \dots x_n), f_2(x_1, x_2, \dots x_n), \dots f_m(x_1, x_2, \dots x_n))$$

$$J_F(x_1, x_2, \dots x_n) \text{ or } \frac{\partial(f_1, f_2, \dots f_m)}{\partial(x_1, x_2, \dots x_n)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$F(\mathbf{x} + \Delta\mathbf{x}) \approx F(\mathbf{x}) + J_F(\mathbf{x})\Delta\mathbf{x}$$

## Jacobian matrix

$$F : \mathbf{R} \times [0, \pi] \times [0, 2\pi] \rightarrow \mathbf{R}^3 \qquad t = r \sin\phi \cos\theta$$

$$F(r, \phi, \theta) = (t, u, v) \qquad u = r \sin\phi \sin\theta$$

$$v = r \cos\phi$$

$$J_F(r, \phi, \theta) = \begin{bmatrix} \frac{\partial t}{\partial r} & \frac{\partial t}{\partial \phi} & \frac{\partial t}{\partial \theta} \\ \frac{\partial u}{\partial r} & \frac{\partial u}{\partial \phi} & \frac{\partial u}{\partial \theta} \\ \frac{\partial v}{\partial r} & \frac{\partial v}{\partial \phi} & \frac{\partial v}{\partial \theta} \end{bmatrix}$$

$$= \begin{bmatrix} \sin\phi \cos\theta & r \cos\phi \cos\theta & -r \sin\phi \sin\theta \\ \sin\phi \sin\theta & r \cos\phi \sin\theta & r \sin\phi \cos\theta \\ \cos\phi & -r \sin\phi & 0 \end{bmatrix}$$

## Parametric model

$$\sum_{\mathbf{x}} \Big( \underbrace{\boxed{I(\mathbf{W}(\mathbf{x};\mathbf{p}))}}_{\text{warped image}} + \underbrace{\boxed{\nabla I}}_{\text{image gradient}} \underbrace{\boxed{\frac{\partial \mathbf{W}}{\partial \mathbf{p}}}}_{\textit{Jacobian}\text{ of the warp}} \Delta\mathbf{p} - \underbrace{\boxed{T(\mathbf{x})}}_{\text{target image}} \Big)^2$$

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

## Jacobian of the warp

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

For example, for affine

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

$$\Rightarrow \frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{array}{c} \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix} \\ \begin{matrix} d_{xx} & d_{yx} & d_{xy} & d_{yy} & d_x & d_y \end{matrix} \end{array}$$

## Parametric model

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

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

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

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

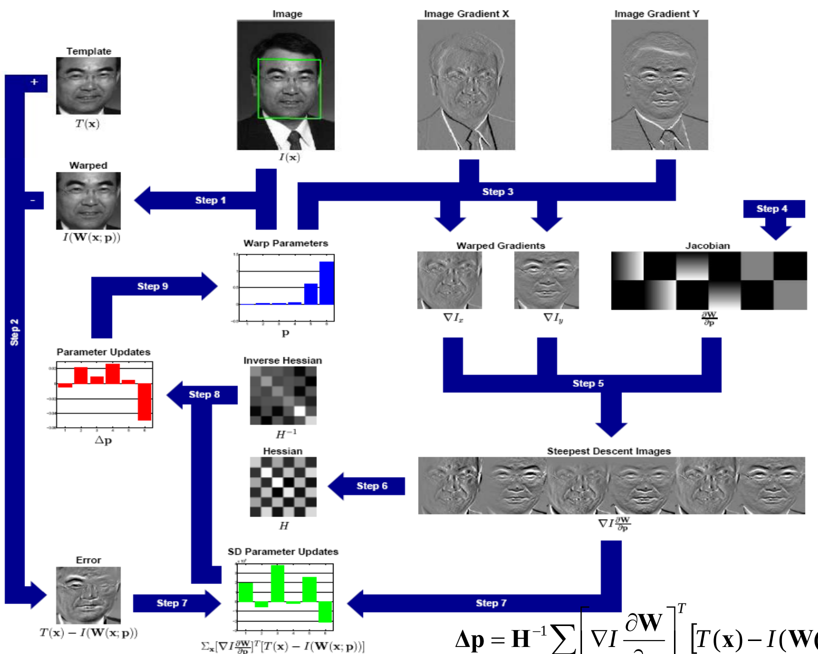
## Lucas-Kanade algorithm

iterate

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

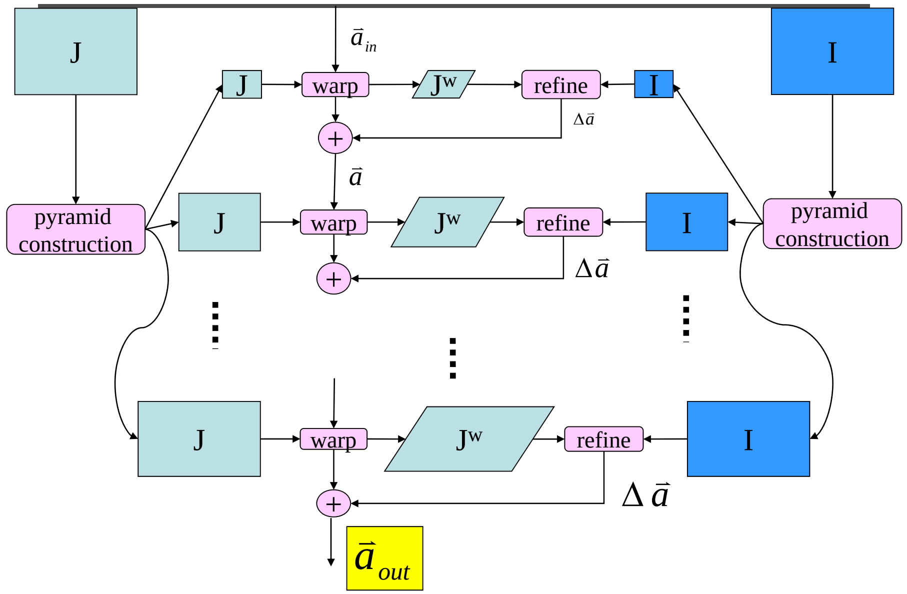
until converge

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




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

## Coarse-to-fine strategy



## Application of image alignment

## Direct vs feature-based

- Direct methods use all information and can be very accurate, but they depend on the fragile “brightness constancy” assumption.
- Iterative approaches require initialization.
- Not robust to illumination change and noise images.
- In early days, direct method is better.

- Feature based methods are now more robust and potentially faster.
- Even better, it can recognize panorama without initialization.

# Tracking

## Tracking

$$I(x,y,t) \xrightarrow{(u,\,v)} I(x+u,y+v,t+1)$$


## Tracking

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

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

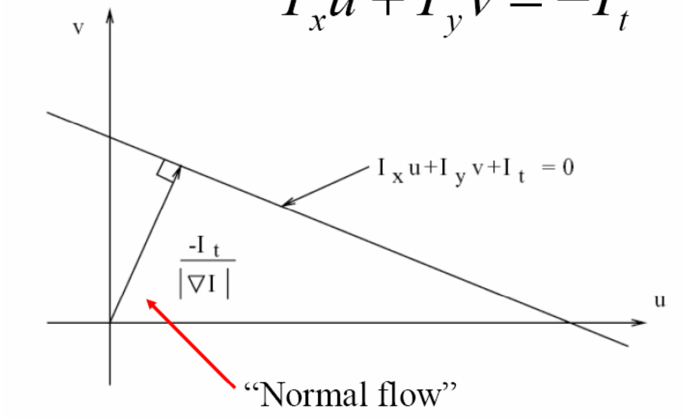
$$uI_x(x,y,t) + vI_y(x,y,t) + I_t(x,y,t) = 0$$

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

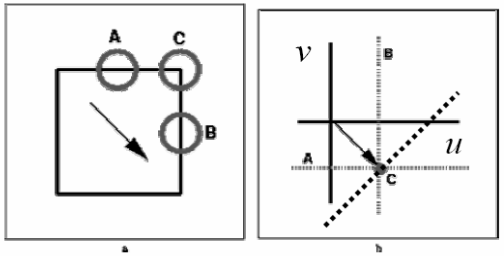
## Optical flow constraint equation

At a single image pixel, we get a line:

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



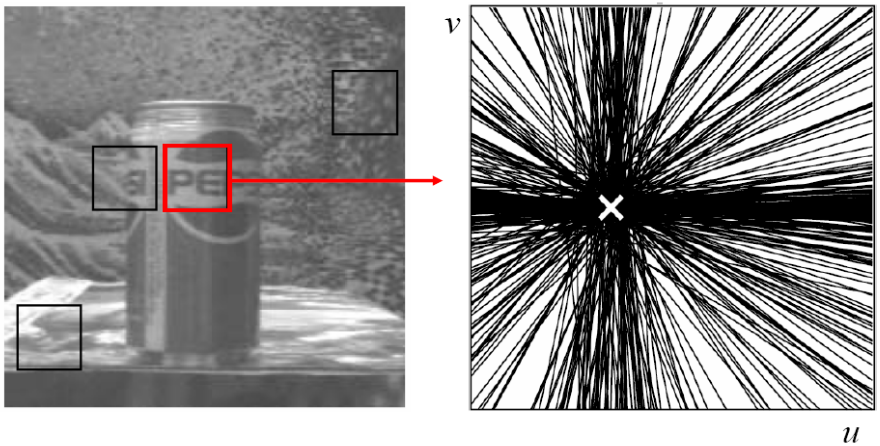

## Multiple constraints




Combine constraints to get an estimate of velocity.

## Area-based method

- Assume spatial smoothness



## 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**

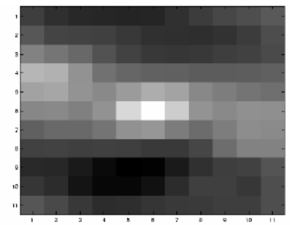
## Area-based method

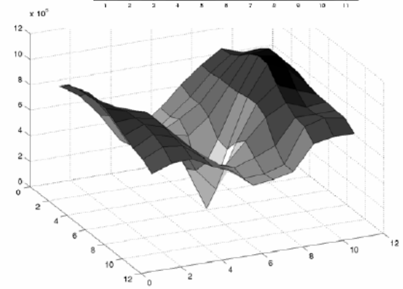
- The eigenvalues tell us about the local image structure.
- They also tell us how well we can estimate the flow in both directions.
- Link to Harris corner detector.

## Textured area

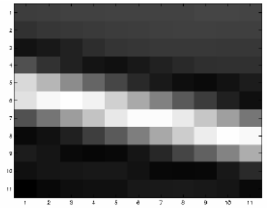
$$\begin{bmatrix}\sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2\end{bmatrix}$$

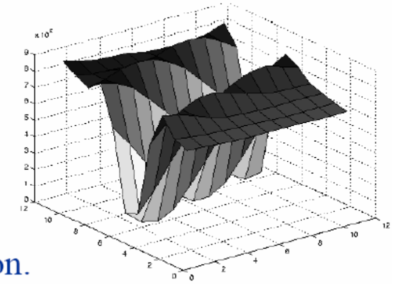
Gradients in $x$ and $y$.





## Edge

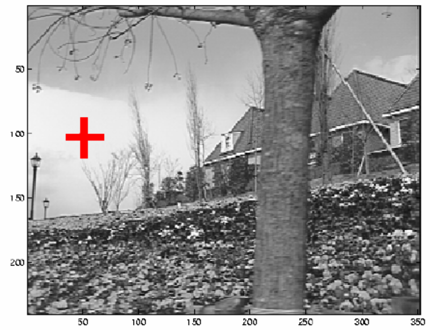


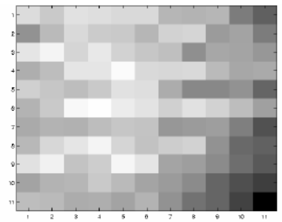


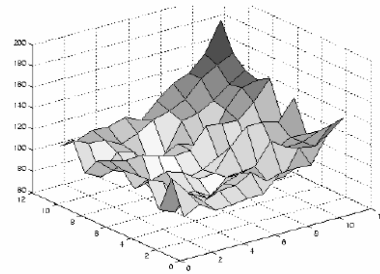
$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}$$

Gradients oriented in one direction.

## Homogenous area





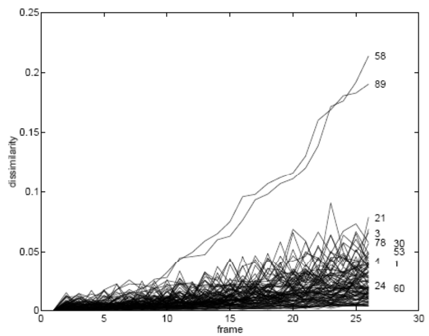


$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}$$

Weak gradients everywhere.

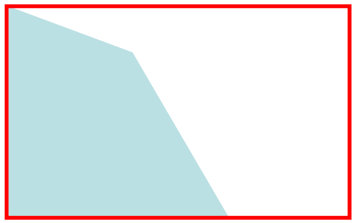
## KLT tracking

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

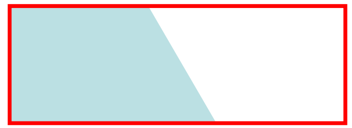



## Aperture problem

## Aperture problem



## Aperture problem



## Demo for aperture problem

- http://www.sandlotscience.com/Distortions/Breathing_Square.htm
- http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm

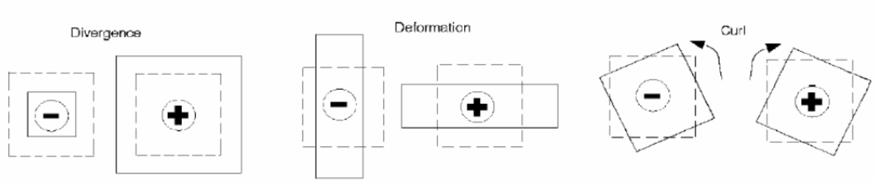
## Aperture problem

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

## 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

DigiVFX


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

## KLT tracking

DigiVFX


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

## SIFT tracking (matching actually)

Frame 0 → Frame 10

## SIFT tracking

Frame 0 → Frame 100

## SIFT tracking

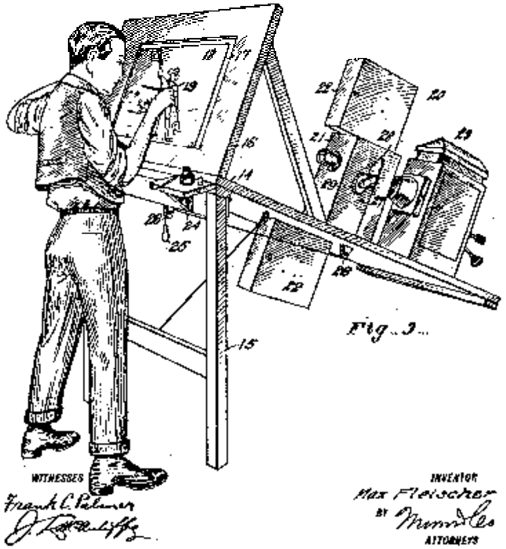
Frame 0 → Frame 200

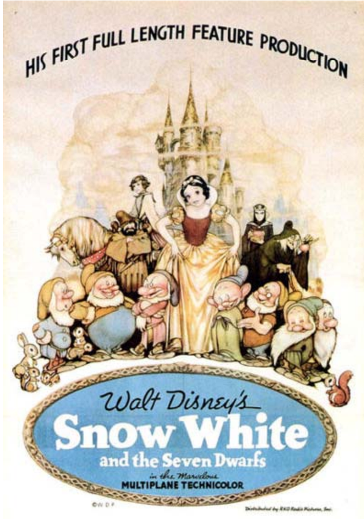
## KLT vs SIFT tracking

- KLT has larger accumulating error; partly because our KLT implementation doesn't have affine transformation?
- SIFT is surprisingly robust
- Combination of SIFT and KLT (example)

http://www.frc.ri.cmu.edu/projects/buzzard/smalls/

## Rotoscoping (Max Fleischer 1914)

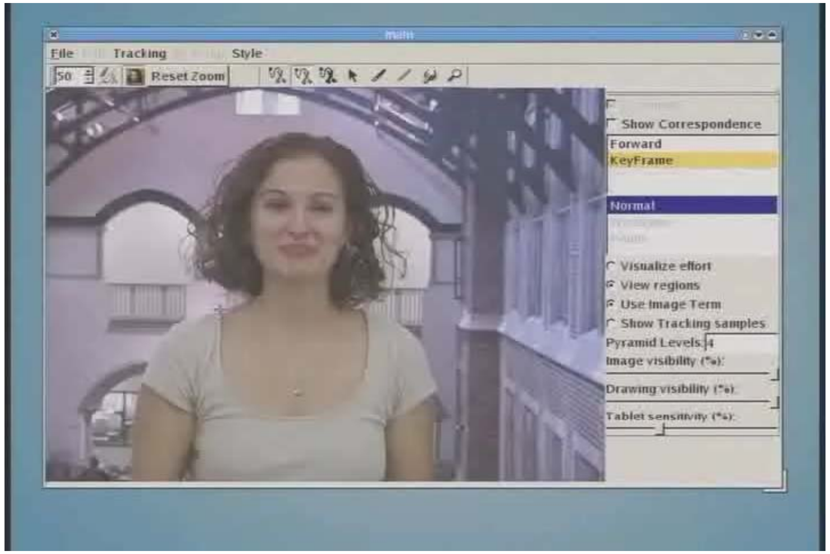



1937



## Tracking for rotoscoping

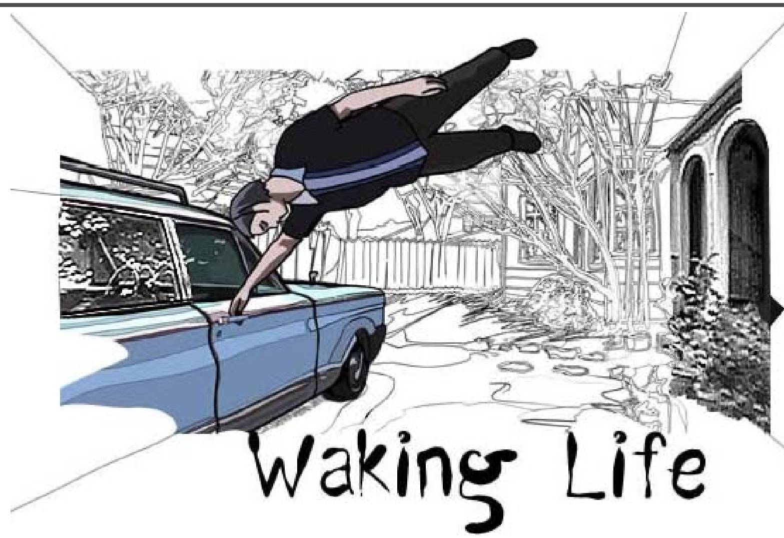





## Tracking for rotoscoping



## Waking life (2001)



## A Scanner Darkly (2006)

- Rotoshop, a proprietary software. Each minute of animation required 500 hours of work.

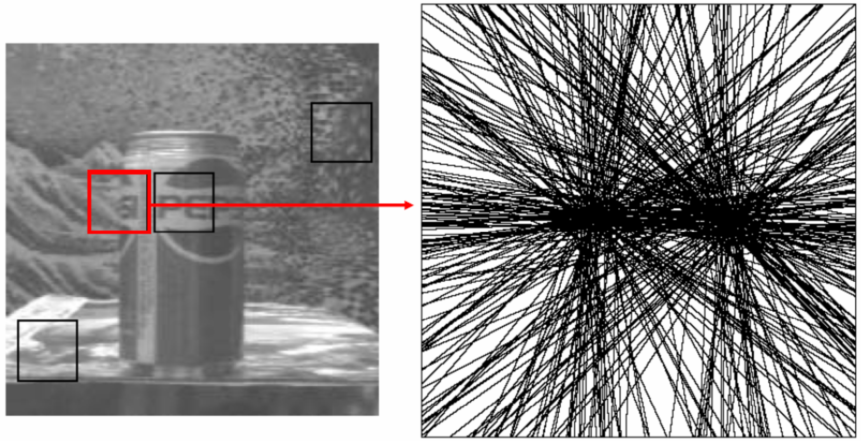
# Optical flow

## Single-motion assumption

Violated by

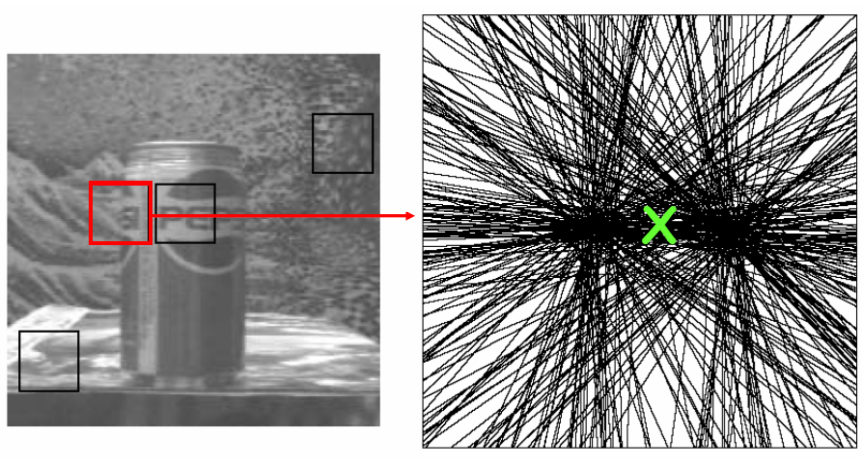
- Motion discontinuity
- Shadows
- Transparency
- Specular reflection
- …

## Multiple motion



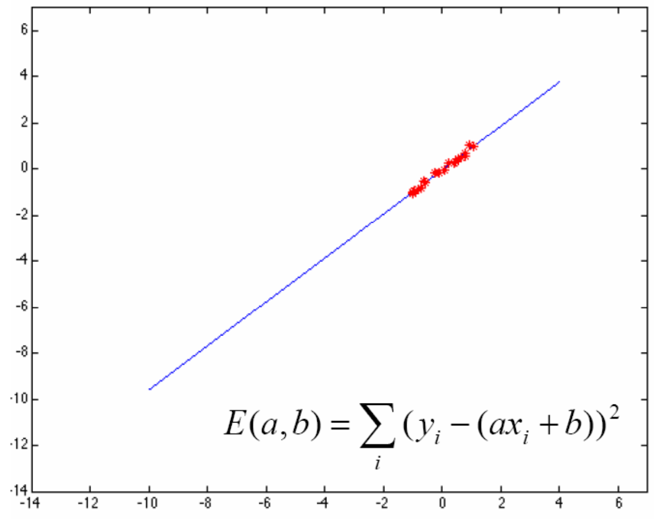
What is the "best" fitting translational motion?

## Multiple motion



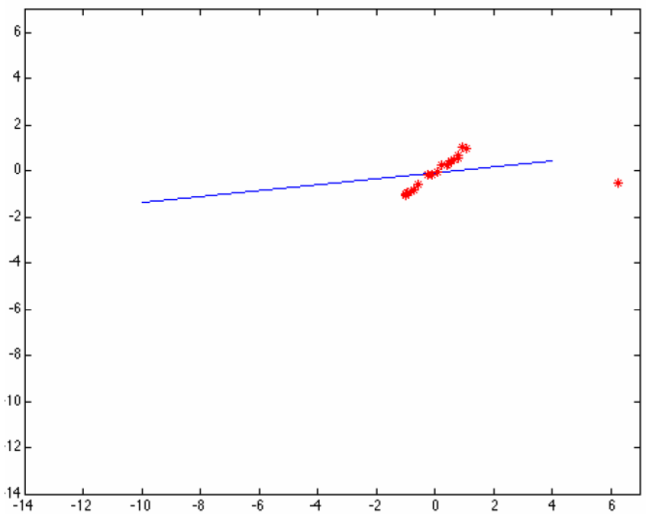
Least squares fit.

## Simple problem: fit a line

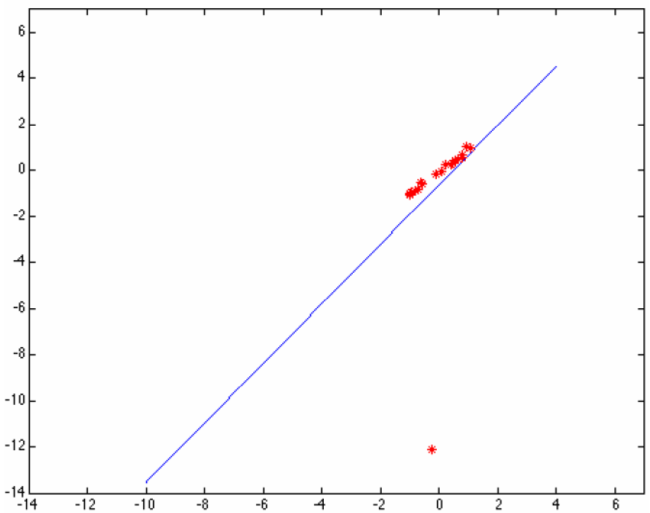


$$E(a,b)=\sum_i (y_i-(ax_i+b))^2$$

## Least-square fit



## Least-square fit

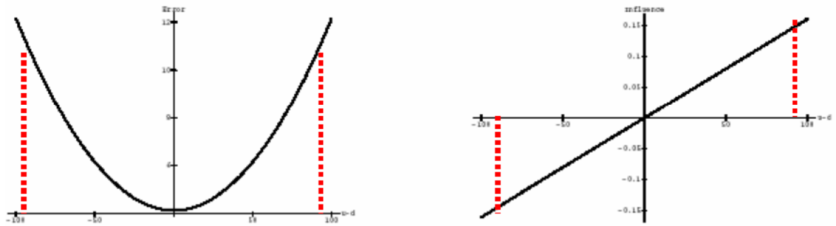


## Robust statistics

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

## Approach

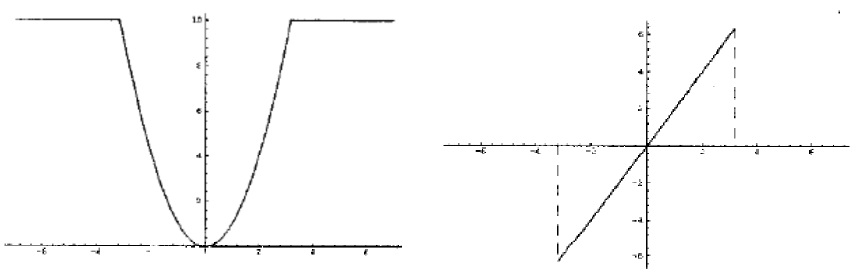
Influence is proportional to the derivative of the $\rho$ function.



Want to give less influence to points beyond some value.

## Robust weighting

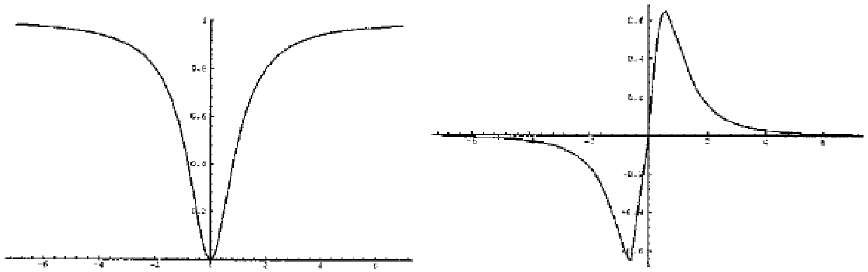
$$\rho(x,\alpha,\lambda) = \begin{cases} \lambda x^2 & \text{if } |x| < \frac{\sqrt{\alpha}}{\sqrt{\lambda}}, \\ \alpha & \text{otherwise.} \end{cases} \quad \psi(x,\alpha,\lambda) = \begin{cases} 2\lambda x & \text{if } |x| < \frac{\sqrt{\alpha}}{\sqrt{\lambda}}, \\ 0 & \text{otherwise.} \end{cases}$$



Truncated quadratic

## Robust weighting

$$\rho(x,\sigma) = \frac{x^2}{\sigma + x^2} \qquad \psi(x,\sigma) = \frac{2x\sigma}{(\sigma + x^2)^2}$$



Geman & McClure

## Robust estimation

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

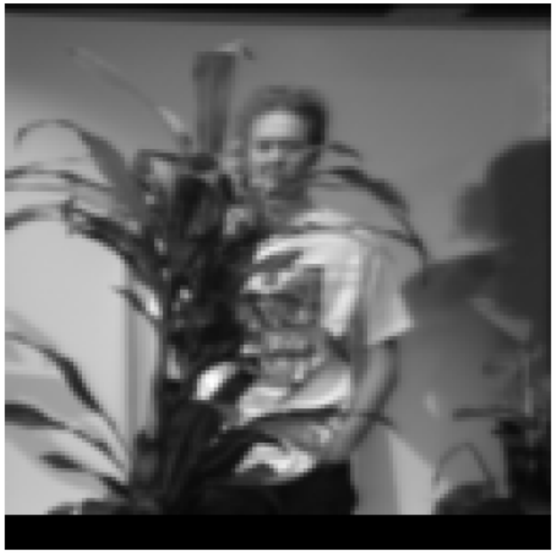
Minimize: differentiate and set equal to zero:

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

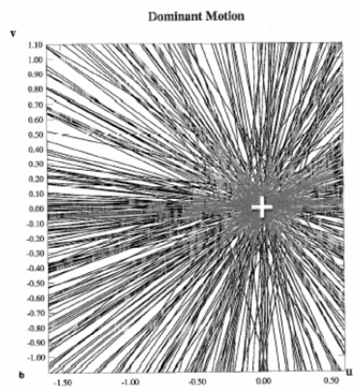
$$\frac{\partial E}{\partial v} = \sum_{x,y\in R} \psi(I_x u + I_y v + I_t, \sigma) I_y = 0$$

*No closed form solution!*

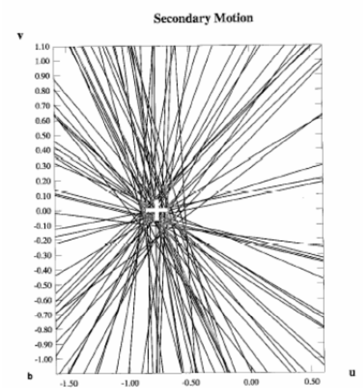
## Fragmented occlusion

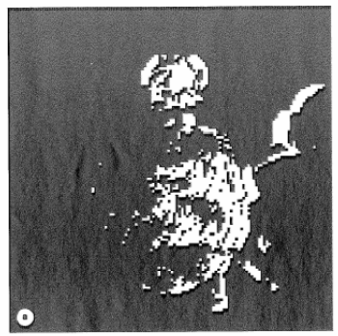


## Results

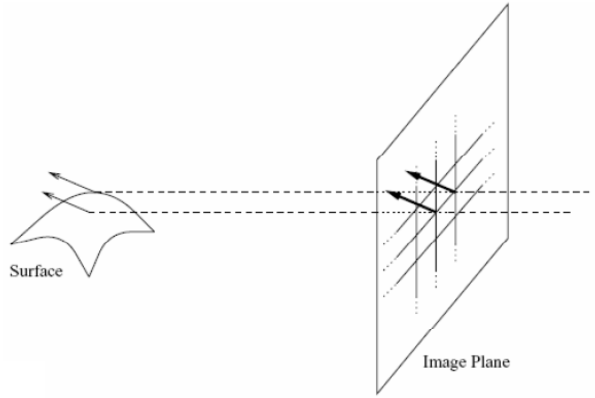



## Results



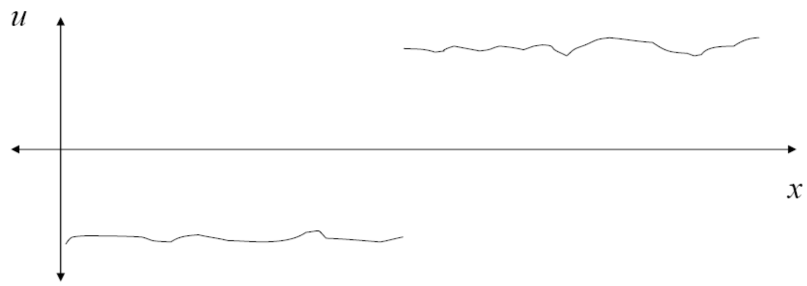
## Regularization and dense optical flow




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

# Formalize this Idea

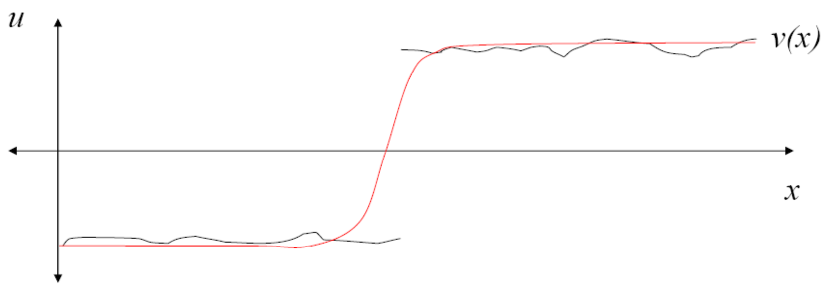
Noisy 1D signal:




Noisy measurements $u(x)$

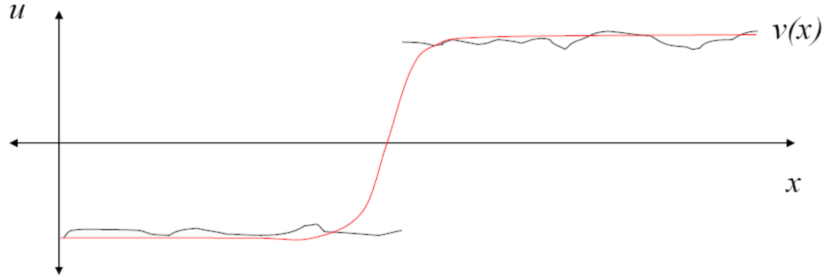
# Regularization

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




Noisy measurements $u(x)$

# Regularization



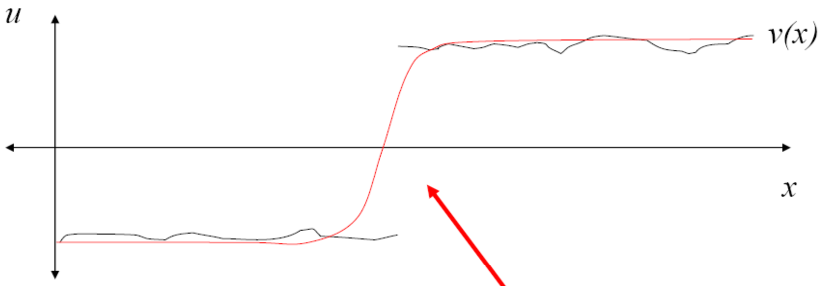

*Minimize:*

Faithful to the data

Spatial smoothness assumption

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

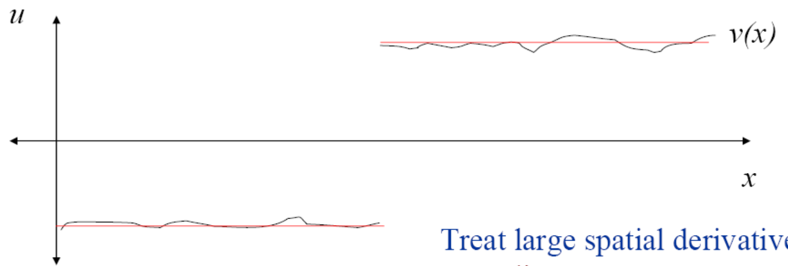
# Discontinuities




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



# Robust Regularization




Treat large spatial derivatives as outliers.

*Minimize:*

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



# “Dense” Optical Flow

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

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

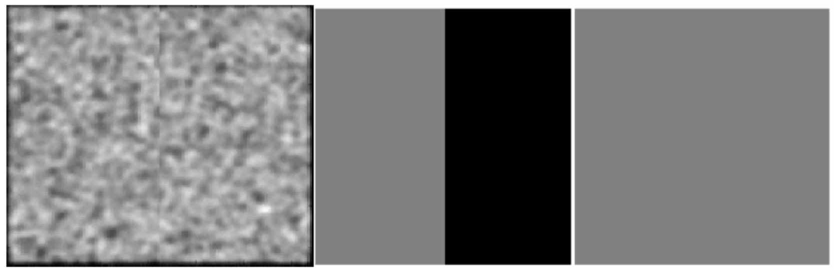
Objective function:

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

When $\rho$ is quadratic = “Horn and Schunck”



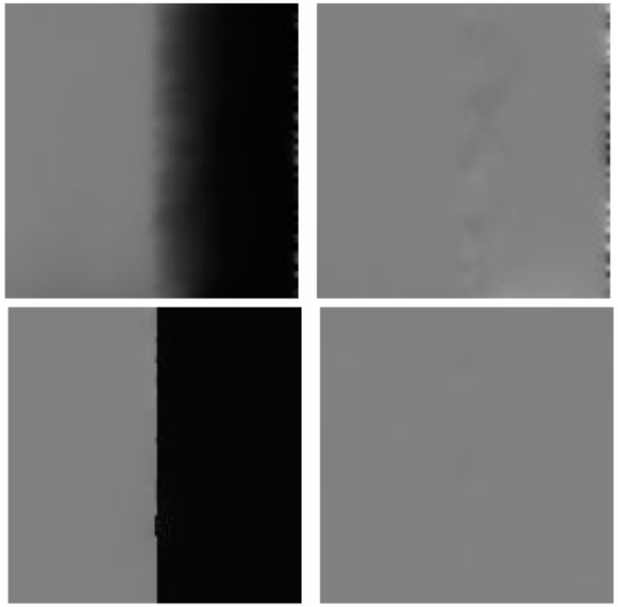
# Example

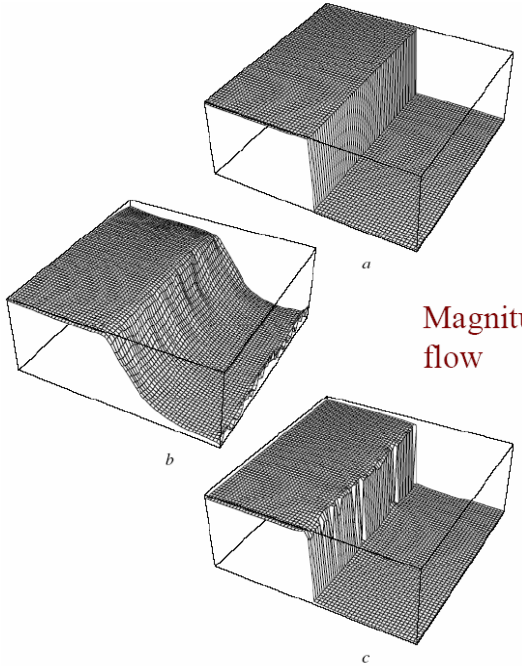


Input image | Horizontal motion | Vertical motion

Quadratic:

Robust:

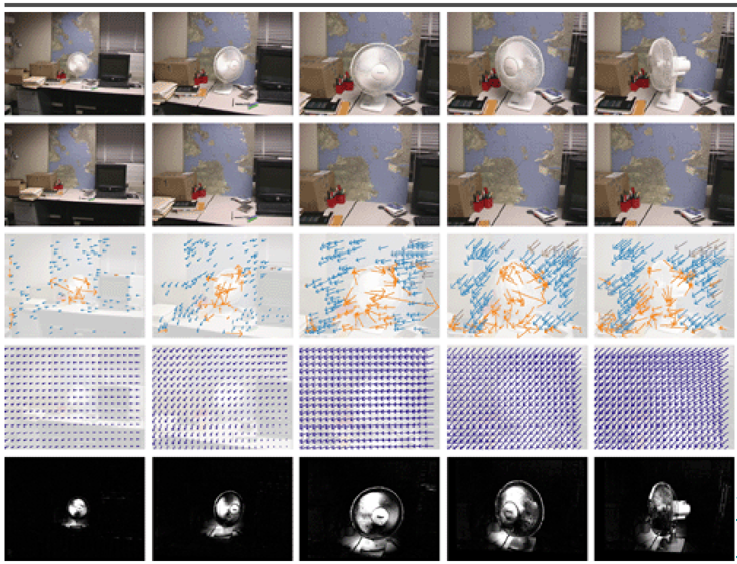





Magnitude of horizontal flow

## Application of optical flow

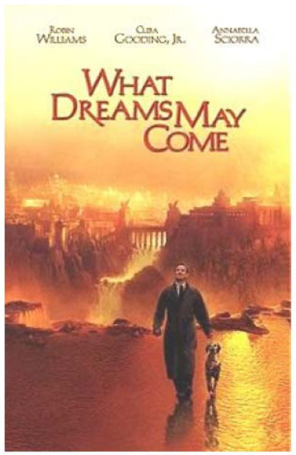




video matching

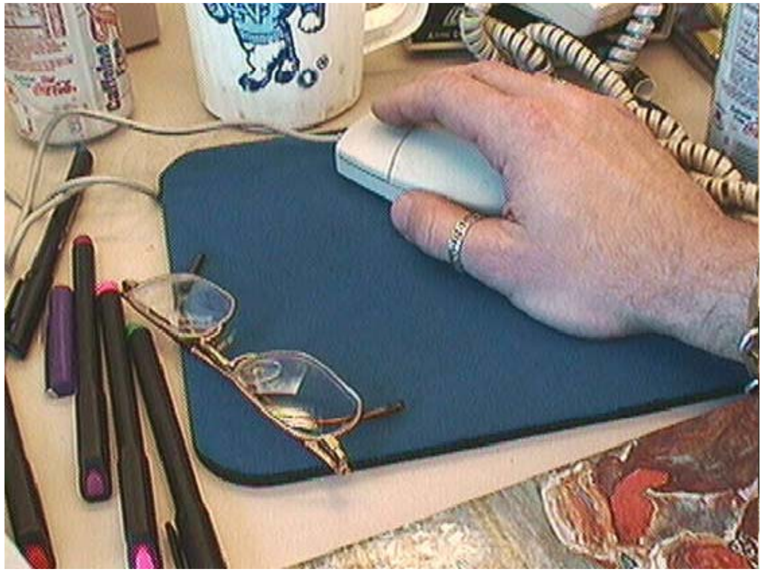
## Applications of Optical Flow






Impressionist effect.
Transfer motion of real world to a painting

## Input for the NPR algorithm



## Brushes

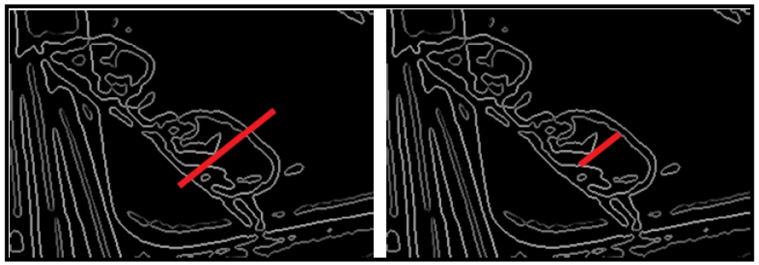
## Edge clipping

## Gradient

## Smooth gradient

## Textured brush

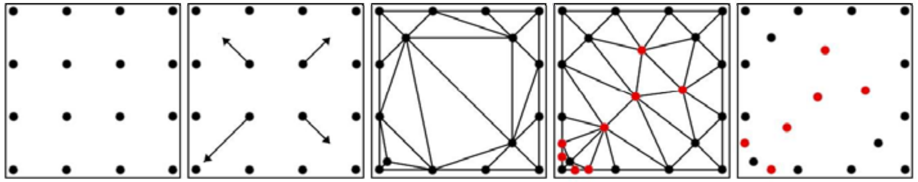
## Edge clipping



## Temporal artifacts

Frame-by-frame application of the NPR algorithm

# Temporal coherence



# References


- 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.