Lecture 4.
Motion I: Optical flow

703142. Computer Vision

Assoz.Prof. Antonio Rodríguez-Sánchez, PhD.

Most slides thanks to Richard Wildes

Outline

- Introduction
- Motion field vs. optical flow
- Brightness constancy
- Gradient-based optical flow estimation
- Finite displacement and feature-based methods
- Object and scene deformations

Introduction

- Time-varying imagery
  - A great deal of useful information can be extracted from time-varying imagery (e.g., video).
    - Temporal image sequences of a dynamic world acquired from a stationary camera.
    - Temporal images sequences of a stationary world acquired from a moving camera.
    - Temporal image sequences of a dynamic world acquired from a moving camera.

Introduction

- Time-varying imagery
  - It might seem foolhardy to consider processing multiple images for extracting information when even one is so challenging.
  - However, multiple images imply additional data on which to base our inferences.
    - Typically, the results are well worth the effort.

Outline

- Introduction
- Motion field vs. optical flow
- Brightness constancy
- Gradient-based optical flow estimation
- Finite displacement and feature-based methods
- Object and scene deformations

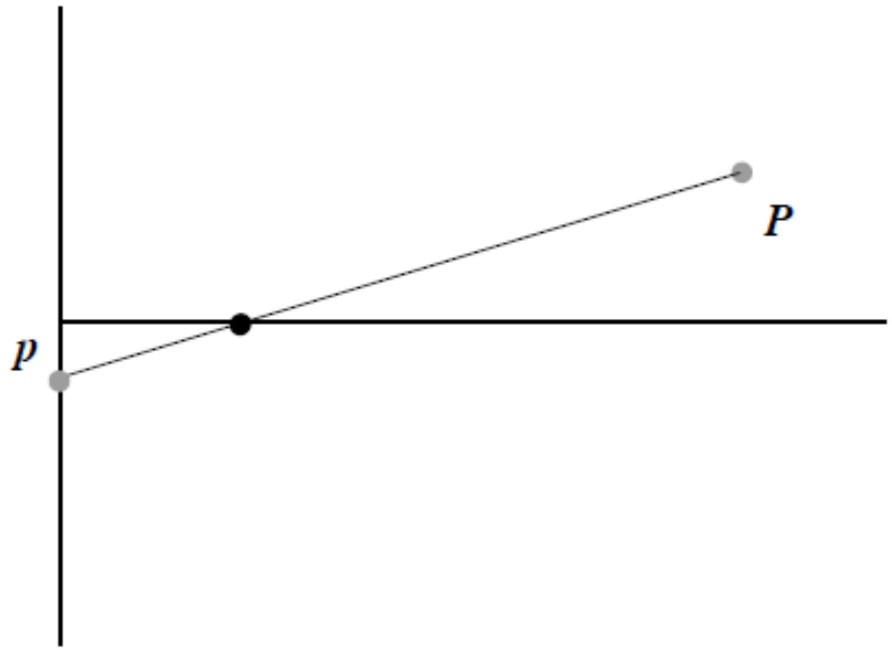
Motion field vs. optical flow

- When objects move in the environment or a camera moves through the environment there are corresponding changes in the images.
- These changes can be used to capture the relative motions as well as the shape of the objects.

Motion field vs. optical flow

- Motion field
  - The motion field assigns a velocity vector to each point in the image according to how the corresponding point in 3D moves.
  - At a particular instance in time a point $\boldsymbol{p}$ in the image corresponds to some point $\boldsymbol{P}$ in the world according to some operative model of image projection,
    - We have

$$\boldsymbol{p} = \Pi(\boldsymbol{P})$$



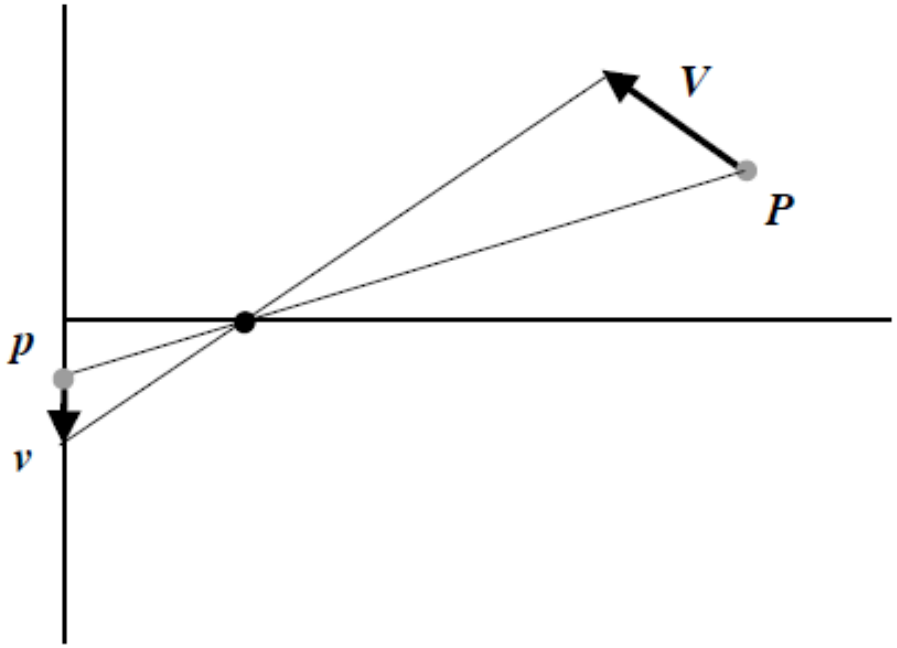
Motion field vs. optical flow

- Motion field
  - At a particular instance in time a point $\boldsymbol{p}$ in the image corresponds to some point $\boldsymbol{P}$ in the world according to some operative model of image projection,
    - We have

$$\boldsymbol{p} = \Pi(\boldsymbol{P})$$

  - Let the point in the world have velocity V relative to the camera, then the image point will have a corresponding velocity, $\boldsymbol{v}$.
    - We have

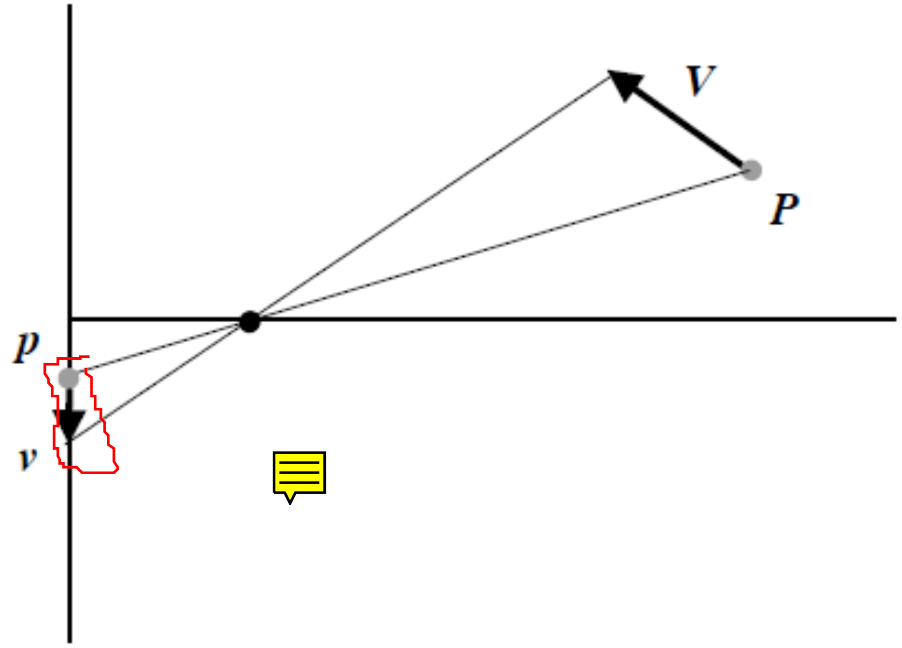
$$\boldsymbol{v} = \frac{d\boldsymbol{p}}{dt} \quad \text{and} \quad \boldsymbol{V} = \frac{d\boldsymbol{P}}{dt}$$



Motion field vs. optical flow

- Motion field
  - At a particular instance in time a point $\boldsymbol{p}$ in the image corresponds to some point $\boldsymbol{P}$ in the world according to some operative model of image projection,
    - We have

$$\boldsymbol{p} = \Pi(\boldsymbol{P})$$



  - Let the point in the world have velocity V relative to the camera, then the image point will have a corresponding velocity, $\boldsymbol{v}$.
    - We have

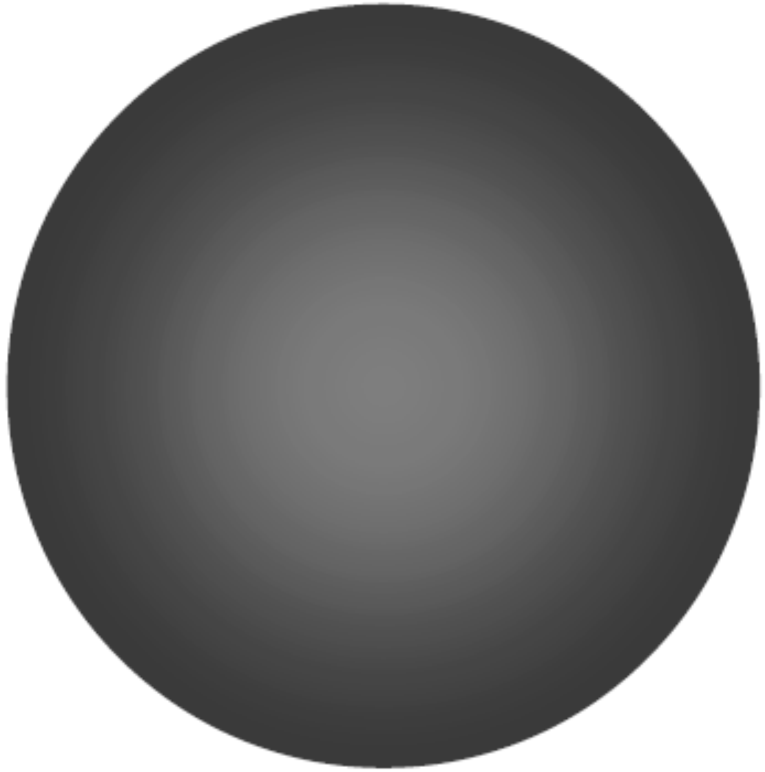
$$\boldsymbol{v} = \frac{d\boldsymbol{p}}{dt} \quad \text{and} \quad \boldsymbol{V} = \frac{d\boldsymbol{P}}{dt} \qquad \frac{d\boldsymbol{p}}{dt} = \frac{d\Pi(\boldsymbol{P})}{dt}$$

Motion field vs. optical flow

- Optical flow
  - Brightness patterns in the image move as the objects in the scene that give rise to them move.
  - Optical flow is the apparent motion of the brightness pattern.
    - The motion that is present in the image.
  - Ideally, the optical flow will correspond to the motion field.
    - But this is not always the case.

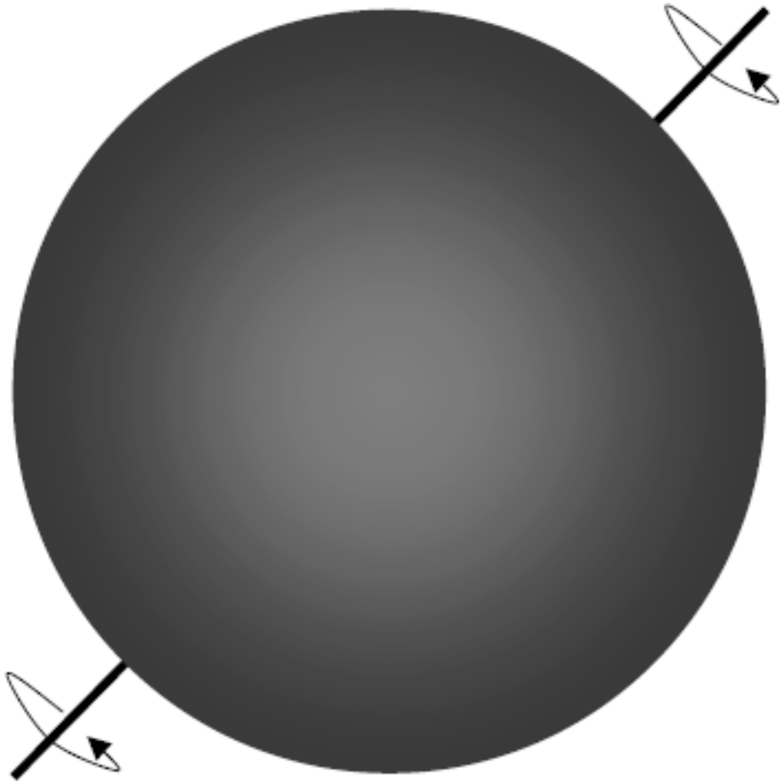
Motion field vs. optical flow

- Rotating sphere
  - Consider perfect sphere in front of a fixed imaging system (camera and illumination).
  - There will be a smooth spatial variation in image brightness (shading) since the surface is curved.



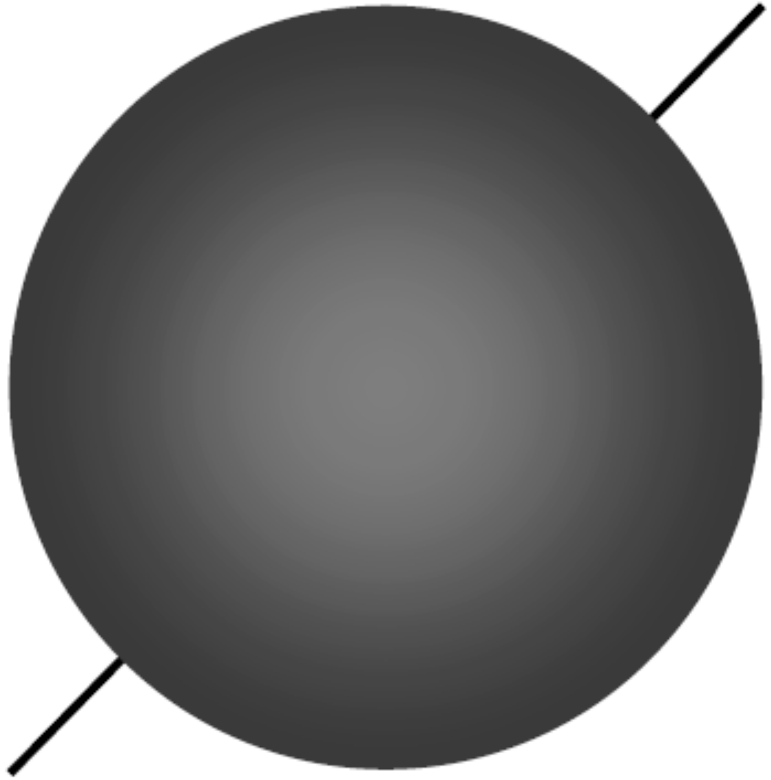
Motion field vs. optical flow

- Rotating sphere
  - Consider perfect sphere in front of a fixed imaging system (camera and illumination).
  - There will be a smooth spatial variation in image brightness (shading) since the surface is curved.
  - Let the sphere rotate.



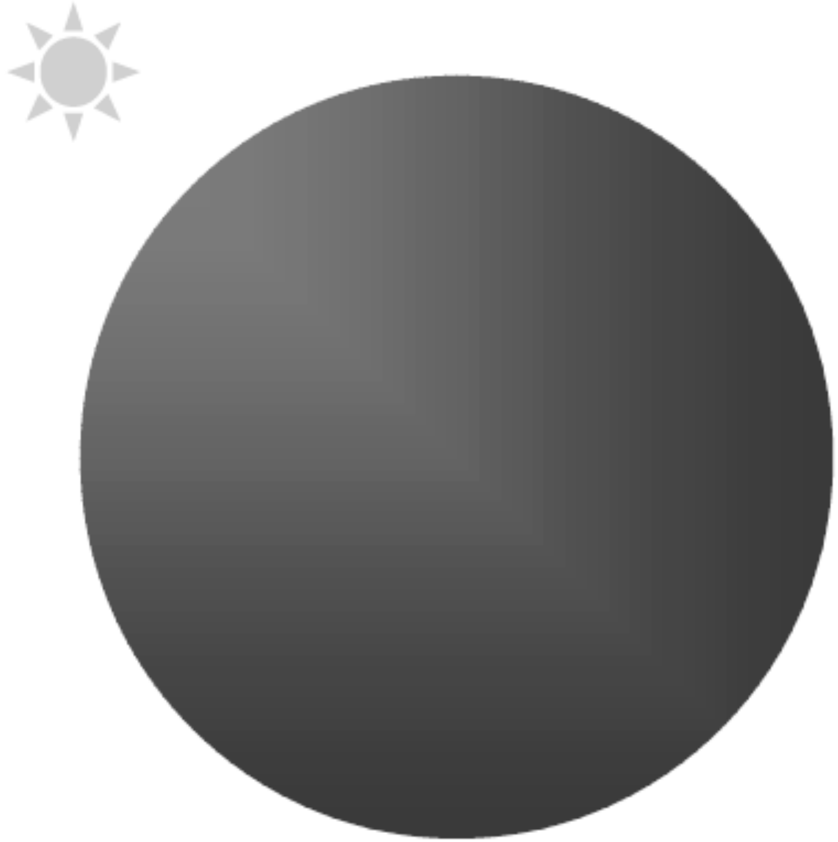
Motion field vs. optical flow

- Rotating sphere
  - Consider perfect sphere in front of a fixed imaging system (camera and illumination).
  - There will be a smooth spatial variation in image brightness (shading) since the surface is curved.
  - Let the sphere rotate.
  - There is no change in the shading pattern.
    - The relationship between the local surface orientation and the imaging system does not vary.
  - The optical flow is zero every where…
  - …despite a nonzero motion field.



Motion field vs. optical flow

- Moving light source
  - Consider a perfect sphere in front of a stationary camera, but moving light source.

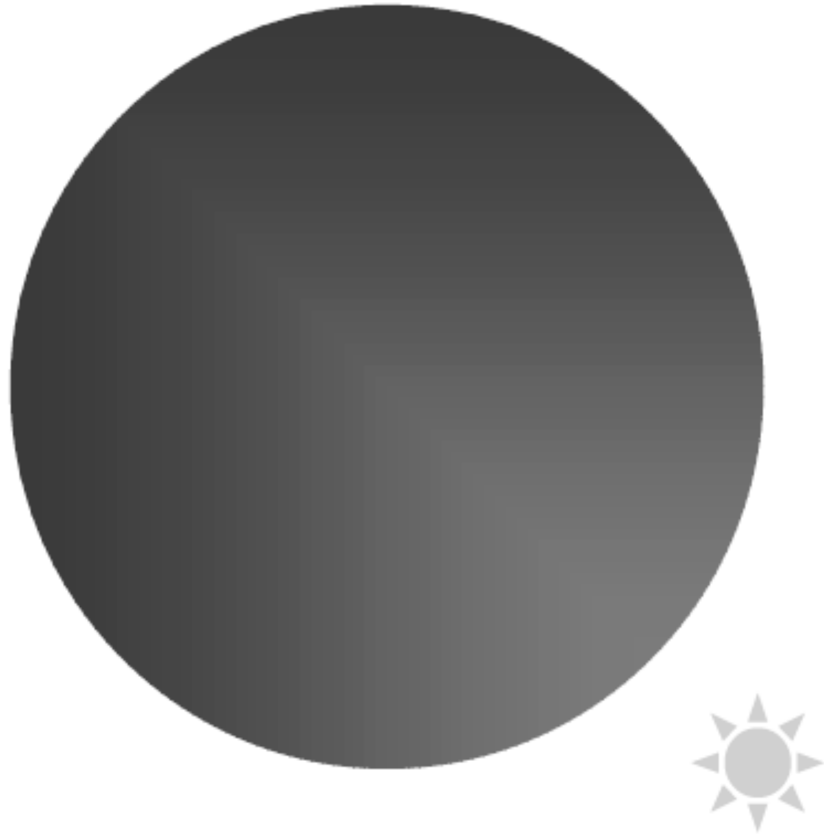


Motion field vs. optical flow

- Moving light source
  - Consider a perfect sphere in front of a stationary camera, but moving light source.

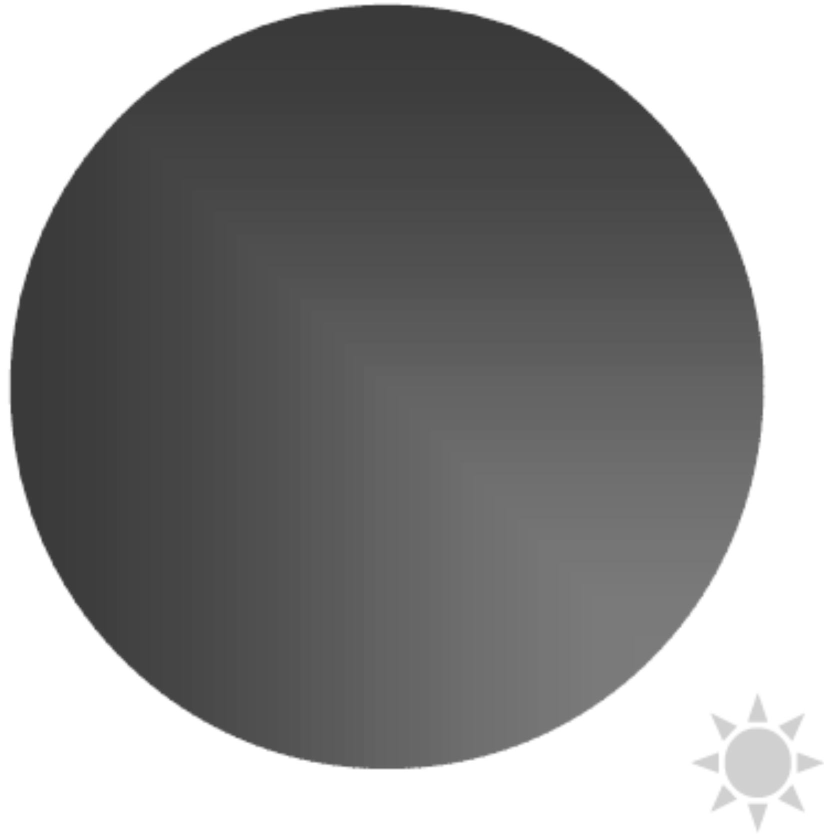
Motion field vs. optical flow

- Moving light source
  - Consider a perfect sphere in front of a stationary camera, but moving light source.



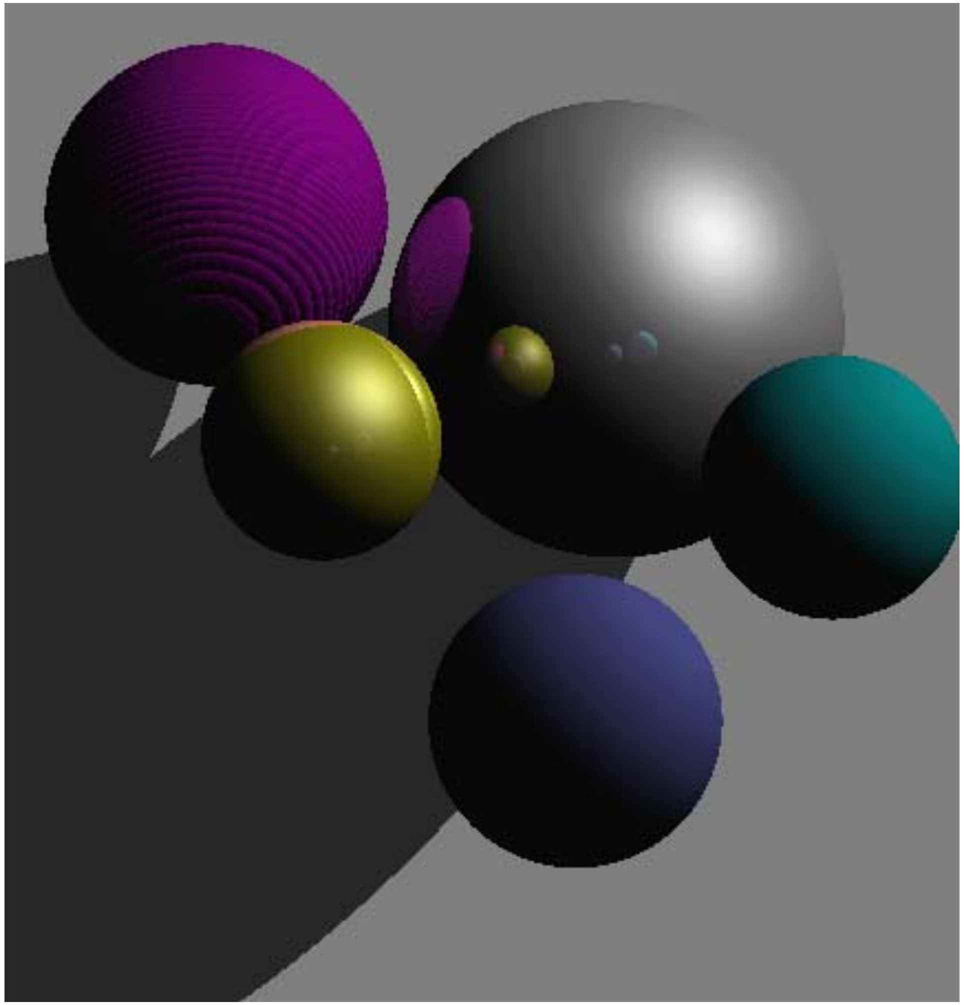
Motion field vs. optical flow

- Moving light source
  - Consider a perfect sphere in front of a stationary camera, but moving light source.
  - Now the shading pattern changes with the variation in source position.
  - The optical flow is nonzero everywhere…
  - …although the motion field is zero.

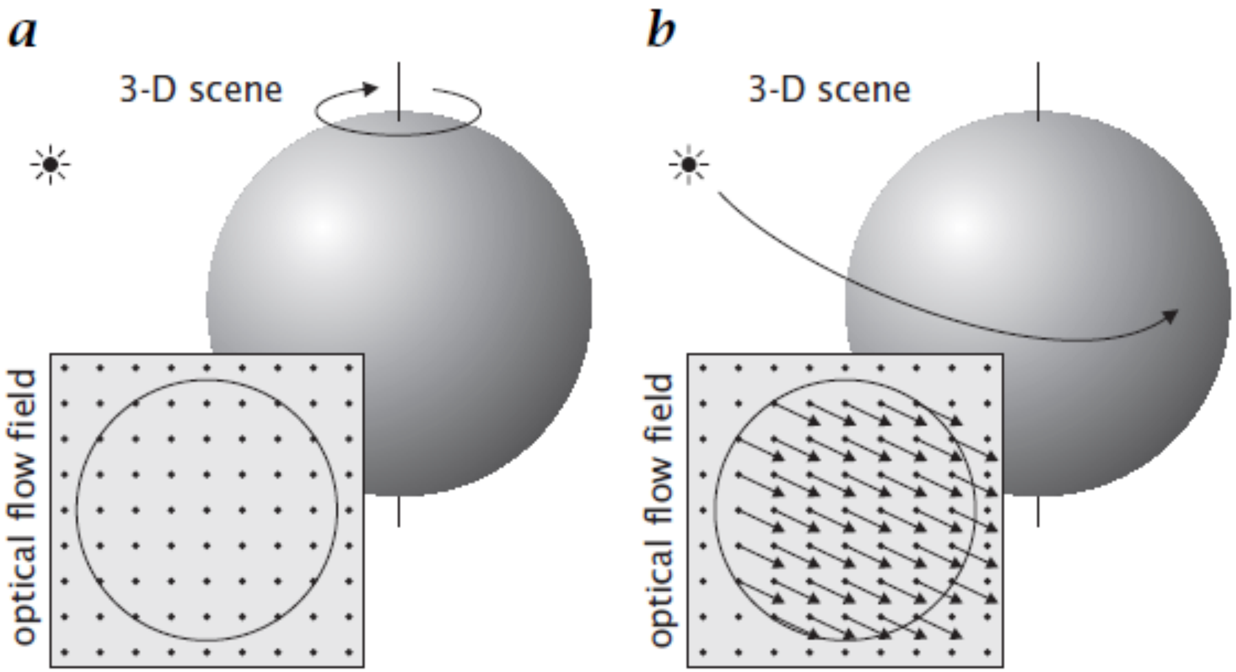


Motion field vs. optical flow

- Moving light source
  - Consider a perfect sphere in front of a stationary camera, but moving light source.
  - Now the shading pattern changes with the variation in source position.
  - The optical flow is nonzero everywhere…
  - …although the motion field is zero.
- Other sources of discrepancy
  - Shadows
  - Specular reflection
  - Virtual images
  - Etc.



Motion field vs. optical flow



Source: Jahne, Computer Vision and applications

Motion field vs. optical flow

- Life is tough, but not too…
  - We are interested in the motion field
    - A purely geometric concept
    - That relates to the structure and dynamics of the scene
  - What we have access to is the optical flow
    - A photometric concept
    - The thing that we can measure in an image.
  - Typically, the motion field and optical flow are in close correspondence
    - But not always
    - As our examples have shown

Outline

- Introduction
- Motion field vs. optical flow
- Brightness constancy
- Gradient-based optical flow estimation
- Finite displacement and feature-based methods
- Object and scene deformations

Brightness constancy

- Relating temporal brightness change to optical flow
  - Let
    - $E(x,y,t)$ be image irradiance at time t and image location (x,y)
    - $u(x,y)$ and $v(x,y)$ be the x and y components of the optical flow, respectively

Brightness constancy

- Relating temporal brightness change to optical flow
  - Let
    - $E(x,y,t)$ be image irradiance at time t and image location (x,y)
    - $u(x,y)$ and $v(x,y)$ be the x and y components of the optical flow, respectively

$$u = \frac{dx}{dt}, v = \frac{dy}{dt}$$

Brightness constancy

- Relating temporal brightness change to optical flow
  - Let
    - $E(x,y,t)$ be image irradiance at time t and image location (x,y)
    - $u(x,y)$ and $v(x,y)$ be the x and y components of the optical flow, respectively

$$u = \frac{dx}{dt}, v = \frac{dy}{dt}$$

  - The brightness constancy assumption is that the irradiance will be the same at time $t+\delta t$ at the point $(x+\delta x, y+\delta y)$
    - $\delta x=u\delta t, \delta y=v\delta t$ for small δt

Brightness constancy

- Relating temporal brightness change to optical flow
  - The brightness constancy assumption is that the irradiance will be the same at time $t+\delta t$ at the point $(x+\delta x, y+\delta y)$
    - $\delta x = u\delta t$, $\delta y = v\delta t$ for small δt

$$E(x+u\delta t, y+v\delta t, t+\delta t) = E(x, y, t)$$

Brightness constancy

- Relating temporal brightness change to optical flow
  - The brightness constancy assumption is that the irradiance will be the same at time $t+\delta t$ at the point $(x+\delta x, y+\delta y)$
    - $\delta x = u\delta t$, $\delta y = v\delta t$ for small δt

$$E(x+u\delta t, y+v\delta t, t+\delta t) = E(x,y,t)$$

  - After some mathematical machinery, we write this as:

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0$$

Brightness constancy

- Relating temporal brightness change to optical flow
  - After some mathematical machinery, we write this as:

$$\frac{\partial E}{\partial x}\frac{dx}{dt}+\frac{\partial E}{\partial y}\frac{dy}{dt}+\frac{\partial E}{\partial t}=0$$

  - Since

$$u=\frac{dx}{dt},v=\frac{dy}{dt}$$

  - Define $E_x=\frac{\partial E}{\partial x}$ $E_y=\frac{\partial E}{\partial y}$ $E_t=\frac{\partial E}{\partial t}$

Brightness constancy

- Relating temporal brightness change to optical flow
  - After some mathematical machinery, we write this as:

$$\frac{\partial E}{\partial x}\frac{dx}{dt}+\frac{\partial E}{\partial y}\frac{dy}{dt}+\frac{\partial E}{\partial t}=0$$

  - Since

$$u=\frac{dx}{dt},v=\frac{dy}{dt}$$

  - Define $E_x=\frac{\partial E}{\partial x}$ $E_y=\frac{\partial E}{\partial y}$ $E_t=\frac{\partial E}{\partial t}$

  - We have the standard form of the **optical flow constraint equation**

$$E_x u+E_y v+E_t=0$$

Brightness constancy

- Relating temporal brightness change to optical flow
  - We have the standard form of the optical flow constraint equation

$$E_x u + E_y v + E_t = 0$$

    - Relates spatial and temporal derivatives of irradiance to optical flow
    - Subject to brightness constancy assumption

Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x u + E_y v + E_t = 0$$

    - Two unknowns of interest

Brightness constancy

- Aperture problem
  – We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y v + E_t = 0$$

    - Two unknowns of interest

Brightness constancy

- Aperture problem
  – We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest

Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained

Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape

Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

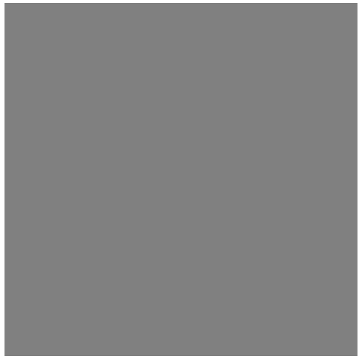
    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape

Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape



Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape

Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

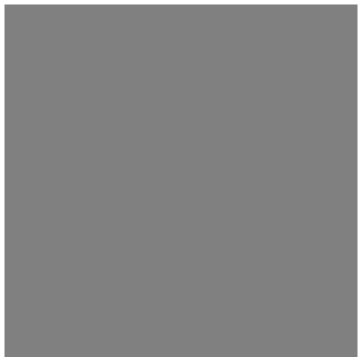
    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape

Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape

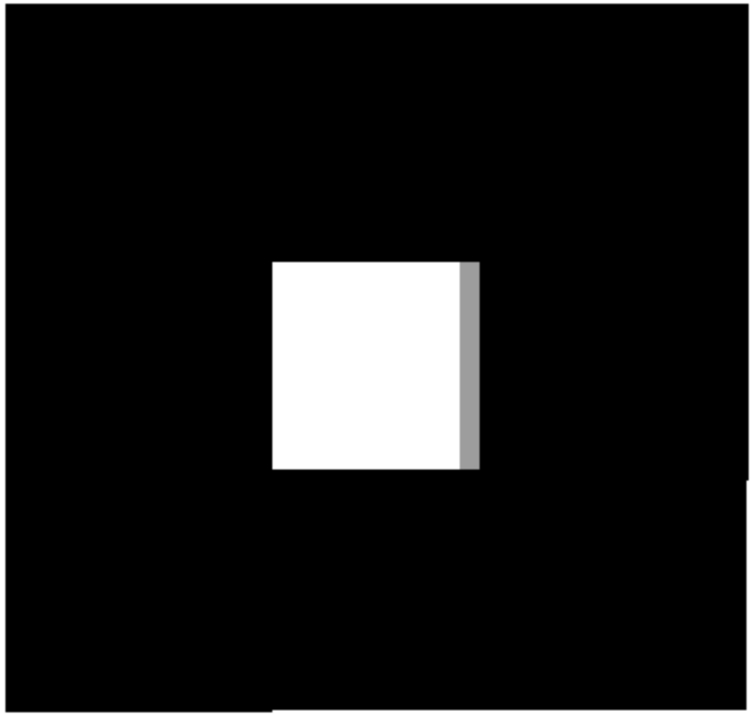


Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape
  - Suppose we restrict consideration to a small region of the image
    - Call this the **aperture**
  - Suppose this aperture is so small that we can see only a single "edge orientation"
    - In the limit the image gradient (E_x, E_y) at a point

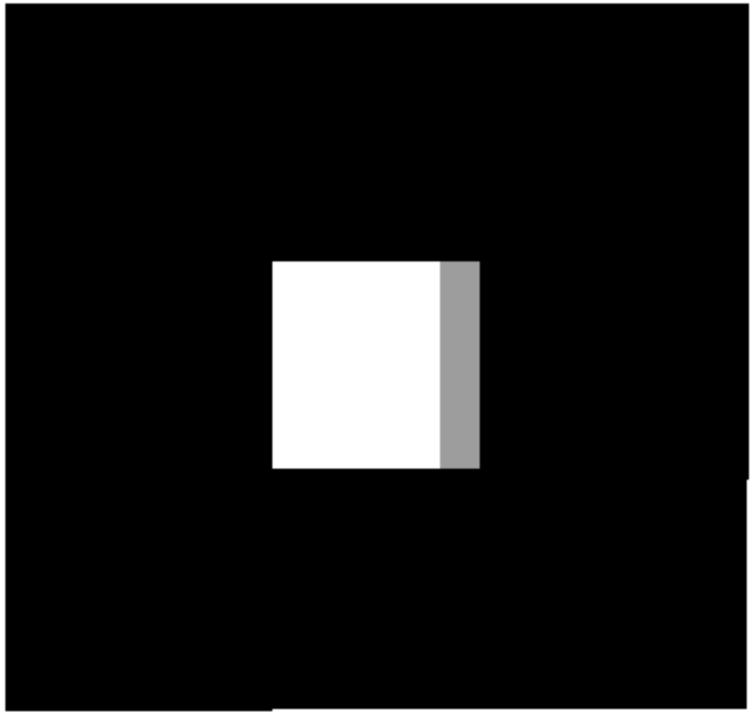


Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape
  - Suppose we restrict consideration to a small region of the image
    - Call this the **aperture**
  - Suppose this aperture is so small that we can see only a single "edge orientation"
    - In the limit the image gradient (E_x, E_y) at a point

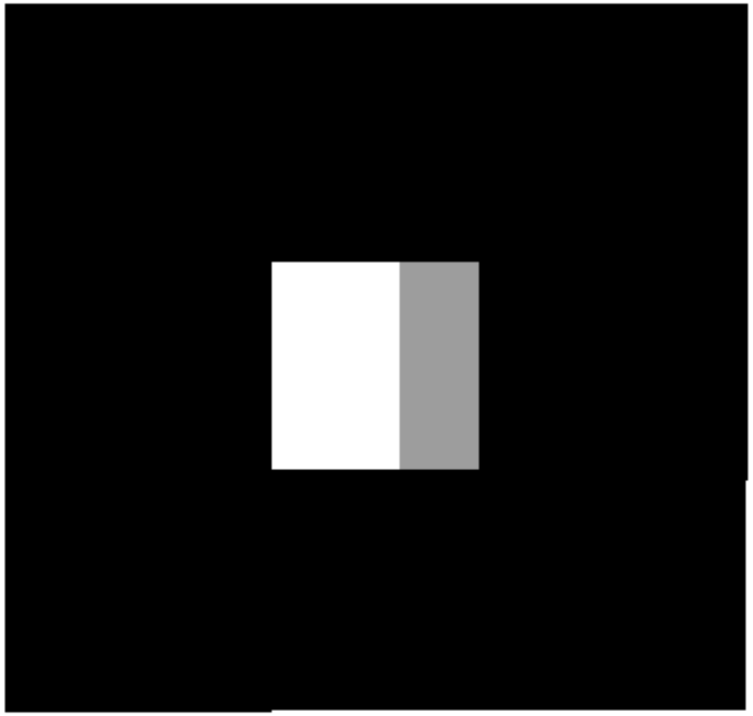


Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape
  - Suppose we restrict consideration to a small region of the image
    - Call this the **aperture**
  - Suppose this aperture is so small that we can see only a single "edge orientation"
    - In the limit the image gradient (E_x, E_y) at a point

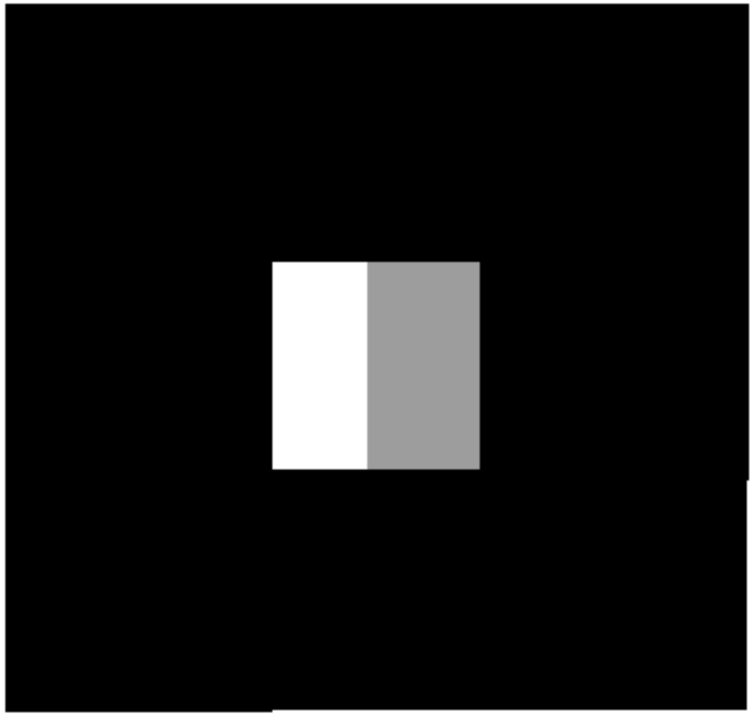


Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape
  - Suppose we restrict consideration to a small region of the image
    - Call this the **aperture**
  - Suppose this aperture is so small that we can see only a single "edge orientation"
    - In the limit the image gradient (E_x,E_y) at a point

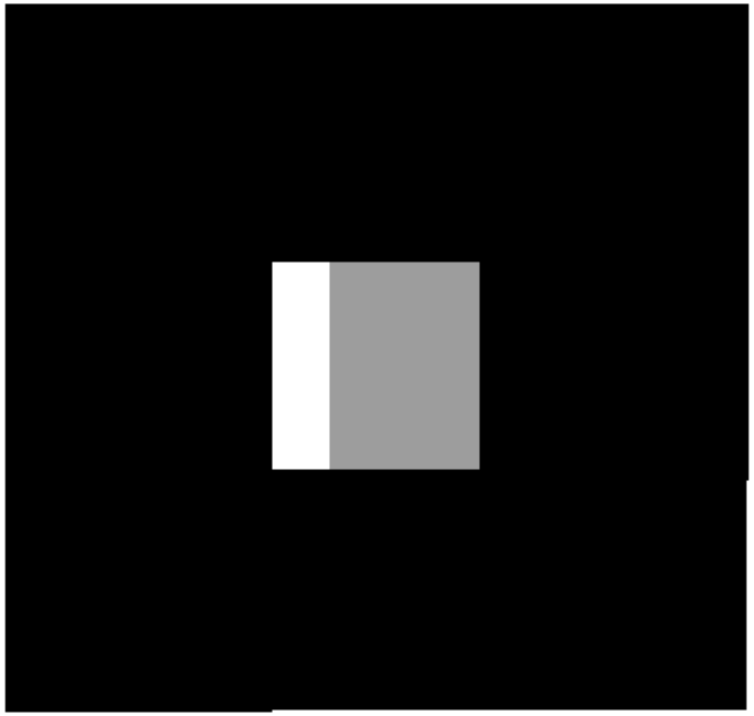


Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape
  - Suppose we restrict consideration to a small region of the image
    - Call this the **aperture**
  - Suppose this aperture is so small that we can see only a single "edge orientation"
    - In the limit the image gradient (E_x, E_y) at a point

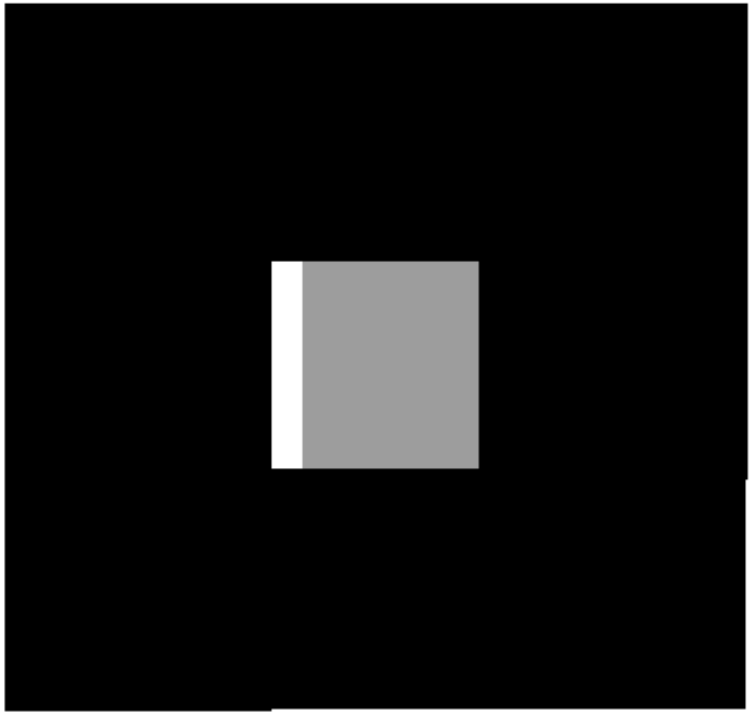


Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - Consider a translating shape
  - Suppose we restrict consideration to a small region of the image
    - Call this the **aperture**
  - Suppose this aperture is so small that we can see only a single “edge orientation”
    - In the limit the image gradient (E_x, E_y) at a point

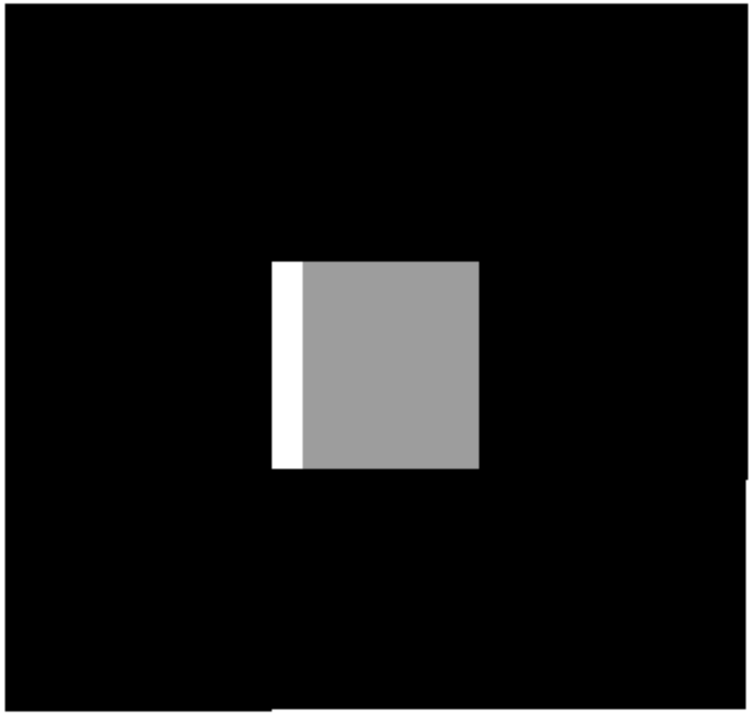


Brightness constancy

- Aperture problem
  - We have the standard form of the optical flow constraint equation

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

    - Two unknowns of interest
    - The solution is under constrained
  - We only have information about the optical flow across the edge, not along the edge.
  - We refer to this limitation as the **aperture problem**.

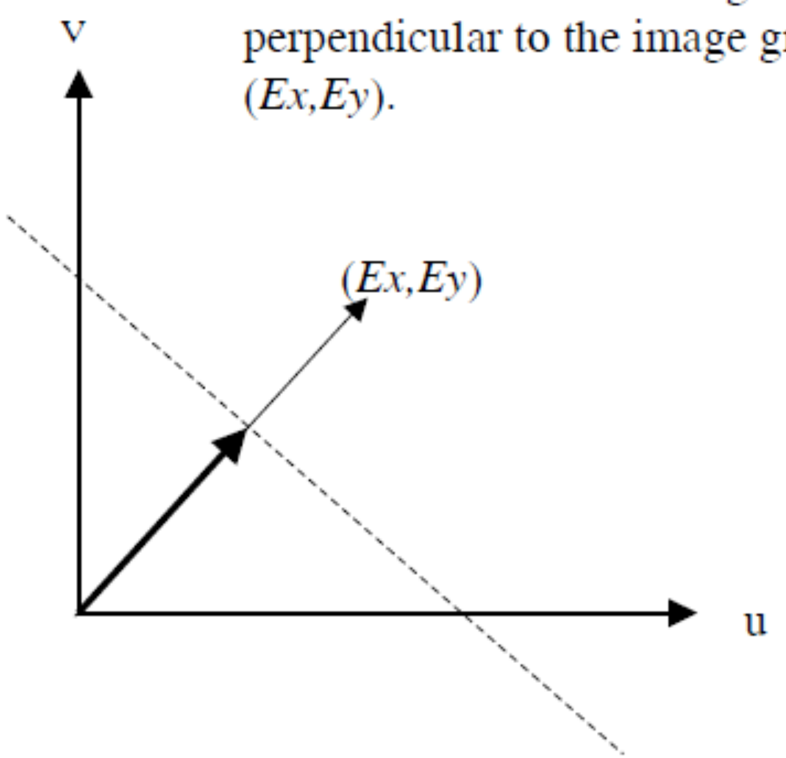


Brightness constancy

- Aperture problem

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

In velocity space, all valid (*u*,*v*) must have their nose along a line perpendicular to the image gradient (*Ex*,*Ey*).

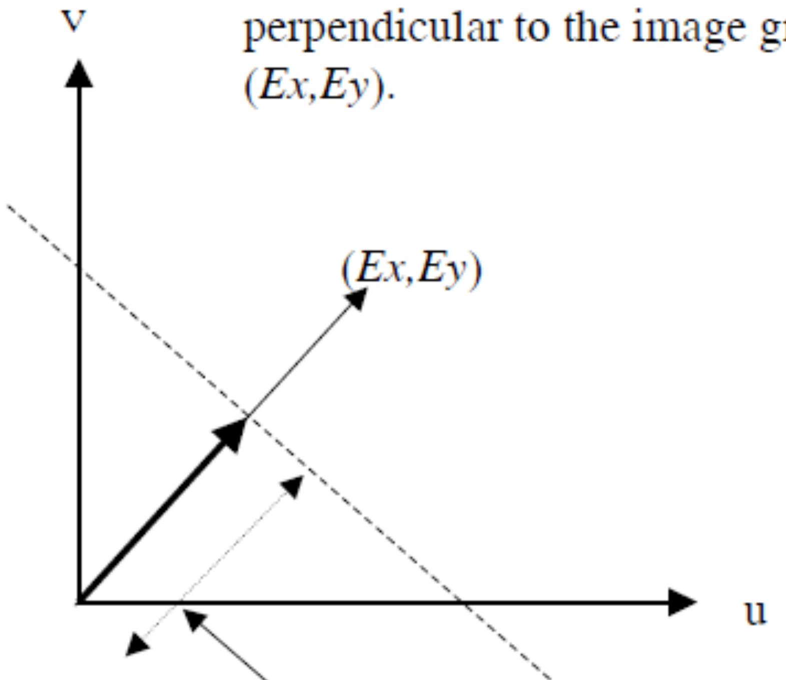


Brightness constancy

- Aperture problem

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

In velocity space, all valid (u,v) must have their nose along a line perpendicular to the image gradient (Ex,Ey).



$$(E_x, E_y) \bullet (u,v) = -E_t$$

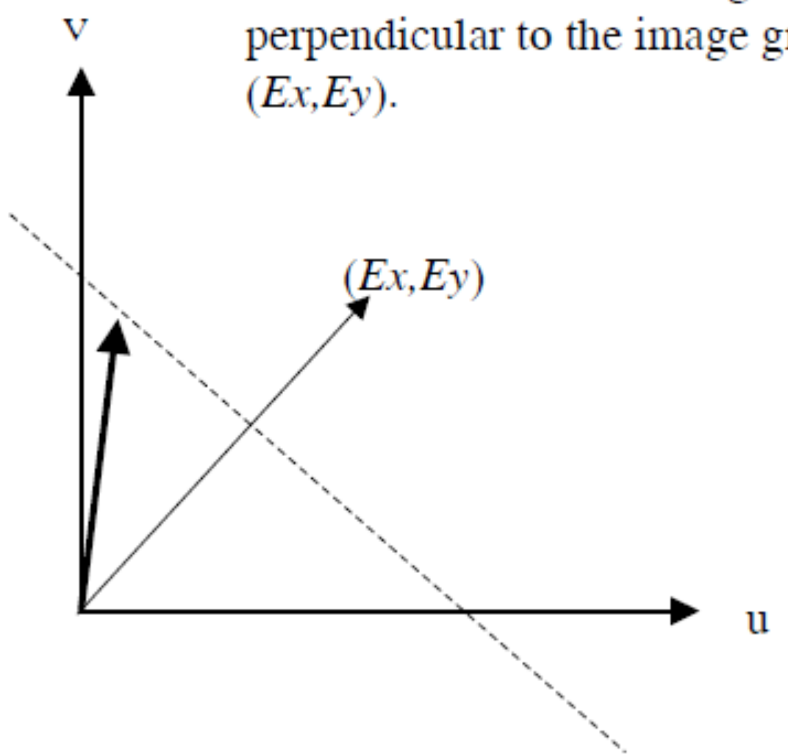
$$\Rightarrow \left\| \frac{-E_t}{\sqrt{E_x^2 + E_y^2}} \right\|$$

Brightness constancy

- Aperture problem

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

In velocity space, all valid (u,v) must have their nose along a line perpendicular to the image gradient (Ex,Ey).

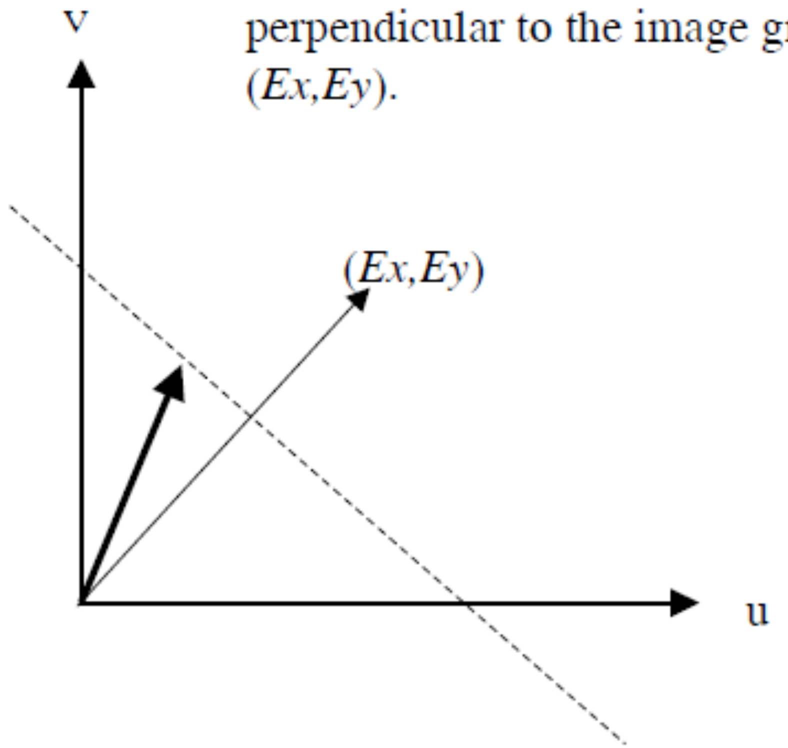


Brightness constancy

- Aperture problem

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

In velocity space, all valid (u,v) must have their nose along a line perpendicular to the image gradient (Ex,Ey).

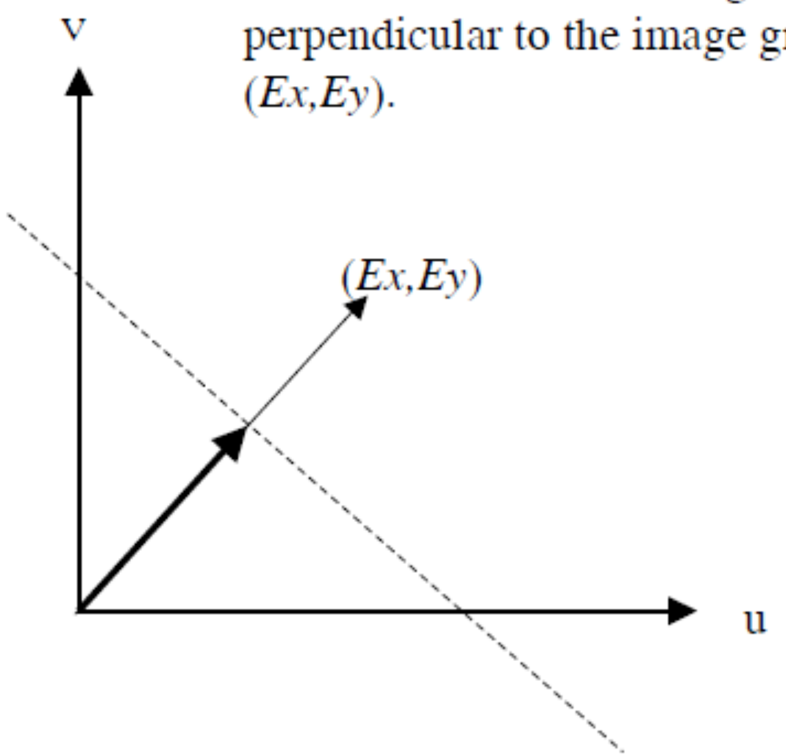


Brightness constancy

- Aperture problem

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

In velocity space, all valid (u,v) must have their nose along a line perpendicular to the image gradient (Ex, Ey).

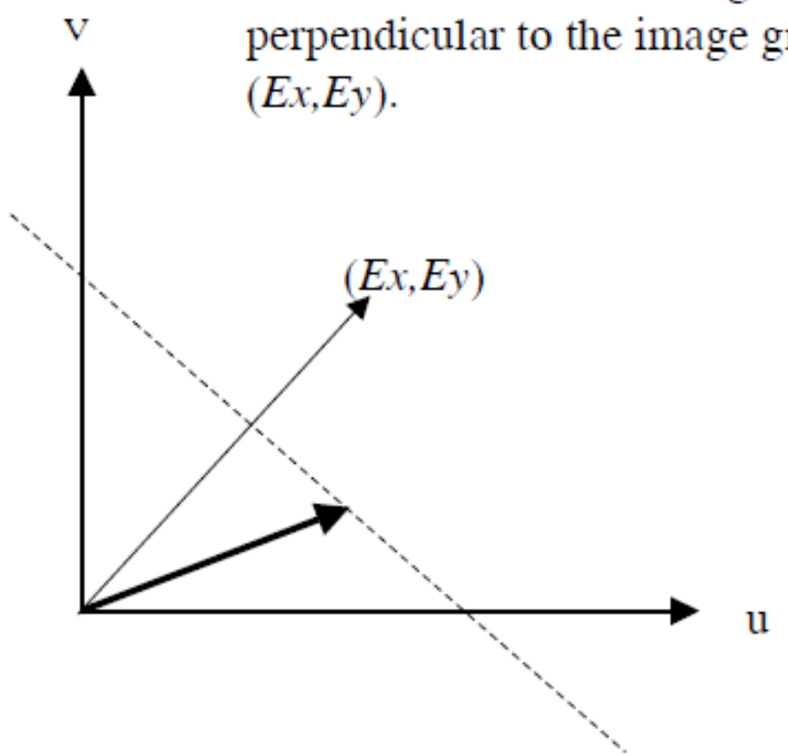


Brightness constancy

- Aperture problem

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

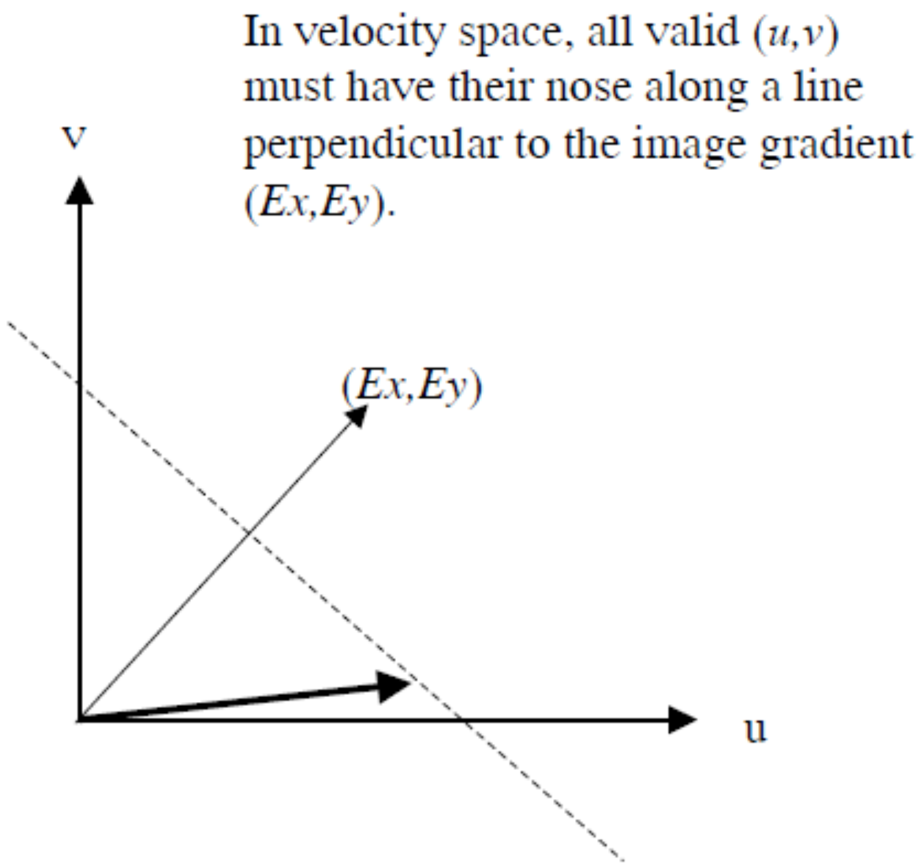
In velocity space, all valid (u,v) must have their nose along a line perpendicular to the image gradient (Ex,Ey).



Brightness constancy

- Aperture problem

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$



Brightness constancy

- In summary
  - We are trying to develop constraints that allow us to recover optical flow from measurements of image irradiance.
  - We have introduced the assumption that a local pattern of image intensity remains constant across an instance of time (brightness constancy).

Brightness constancy

- In summary
  - We are trying to develop constraints that allow us to recover optical flow from measurements of image irradiance.
  - We have introduced the assumption that a local pattern of image intensity remains constant across an instance of time (brightness constancy).
  - This allowed us to derive a fundamental equation that relates derivatives of irradiance to optical flow (the optical flow constraint equation)

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

  - However, there is not enough constraint to unambiguously determine the flow (aperture problem).

Brightness constancy

- In summary
  - We are trying to develop constraints that allow us to recover optical flow from measurements of image irradiance.
  - We have introduced the assumption that a local pattern of image intensity remains constant across an instance of time (brightness constancy).
  - This allowed us to derive a fundamental equation that relates derivatives of irradiance to optical flow (the optical flow constraint equation)

$$E_x \boxed{u} + E_y \boxed{v} + E_t = 0$$

  - However, there is not enough constraint to unambiguously determine the flow (aperture problem).
    - We seek additional constraint to uniquely define the optical flow.
    - To allow for algorithmic recovery.

Outline

- Introduction
- Motion field vs. optical flow
- Brightness constancy
- Gradient-based optical flow estimation
- Finite displacement and feature-based methods
- Object and scene deformations

Gradient optical flow estimation

- We have
  - We have derived the optical flow constraint equation

$$E_x u + E_y v + E_t = 0$$

  - This is not enough to uniquely define the optical flow solution.

Gradient optical flow estimation

- We have
  - We have derived the optical flow constraint equation

$$E_x u + E_y v + E_t = 0$$

  - This is not enough to uniquely define the optical flow solution.
- We need
  - Additional constraint so that at (every image location) we have two equations in two unknowns to define a solution.

Gradient optical flow estimation

- We have
  - We have derived the optical flow constraint equation

$$E_x u + E_y v + E_t = 0$$

  - This is not enough to uniquely define the optical flow solution.
- We need
  - Additional constraint so that at (every image location) we have two equations in two unknowns to define a solution.
  - Several approaches have been developed
  - Differentiate the present constraint equation to generate additional constraint equations
    - Variational smoothness with boundary conditions
    - Assume flow constancy over some finite window

Additional constraints

- Differentiate the present constraint equation to generate additional constraint equations
  - Variational smoothness with boundary conditions
  - Assume flow constancy over some finite window

Global smoothness

- Horn and Schunck (1980)
- The apparent velocity of the brightness pattern varies smoothly almost everywhere in the image.
  - Opaque objects undergo rigid motion or deformation
  - Neighboring points on the objects have similar velocities and the velocity field of the brightness pattern in the image varies smoothly almost everywhere.

Global smoothness

- Horn and Schunck (1980)
- The apparent velocity of the brightness pattern varies smoothly almost everywhere in the image.
  - Opaque objects undergo rigid motion or deformation
  - Neighboring points on the objects have similar velocities and the velocity field of the brightness pattern in the image varies smoothly almost everywhere.
- Additional constraint
  - Minimize the square of the magnitude of the gradient of the optical flow velocity

Global smoothness

- Additional constraint
  - Minimize the square of the magnitude of the gradient of the optical flow velocity

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \quad \text{and} \quad \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 .$$

Global smoothness

- Additional constraint
  - Minimize the square of the magnitude of the gradient of the optical flow velocity

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \quad \text{and} \quad \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 .$$

  - And then, minimize

$$\mathscr{E}_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 .$$

  - Which they define as the departure of smoothness

Additional constraints

- Differentiate the present constraint equation to generate additional constraint equations
  - Variational smoothness with boundary conditions
  - Assume flow constancy over some finite window

Assume flow constancy over window

- Lucas and Kanade (1981)
- Flow constancy assumes that over some window, W, the values of (u,v) are constant.

Assume flow constancy over window

- Lucas and Kanade (1981)
- Flow constancy assumes that over some window, ***W***, the values of ***(u,v)*** are constant.
- We seek to minimize

$$\min_{(u,v)} \sum \sum_{W} (E_x u + E_y v + E_t)^2$$

Assume flow constancy over window

- Lucas and Kanade (1981)
- Flow constancy assumes that over some window, ***W***, the values of ***(u,v)*** are constant.
- We seek to minimize

$$\min_{(u,v)} \sum \sum_{W} (E_x u + E_y v + E_t)^2$$

- We solve this by differentiation and equalizing to 0

$$\sum \sum (E_x u + E_y v + E_t) E_x = 0$$

$$\sum \sum (E_x u + E_y v + E_t) E_y = 0$$

Assume flow constancy over window

- Lucas and Kanade (1981)
- Flow constancy assumes that over some window, ***W***, the values of ***(u,v)*** are constant.
- We seek to minimize

$$\min_{(u,v)} \sum \sum_{W} (E_x u + E_y v + E_t)^2$$

- We solve this by differentiation and equalizing to 0

$$\sum \sum (E_x u + E_y v + E_t) E_x = 0$$
$$\sum \sum (E_x u + E_y v + E_t) E_y = 0$$

$\Rightarrow$ **Two equations with two unknowns !**

Assume flow constancy over window

$$\sum\sum(E_x u + E_y v + E_t)E_x = 0$$
$$\sum\sum(E_x u + E_y v + E_t)E_y = 0$$

Two equations with two unknowns !

- Whose solution is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

Assume flow constancy over window

- Whose solution is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

  - Input: A temporal sequence of two images
  - Output: A pair of optical flow images, a U an V image
  - We compute the equation above and store the recovered ***(u,v)*** in the positions ***(i,j)*** in the U and V images

Assume flow constancy over window

- Whose solution is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

  - Window size
    - Smaller windows give better precision
    - Larger windows provide better performance in presence of noise

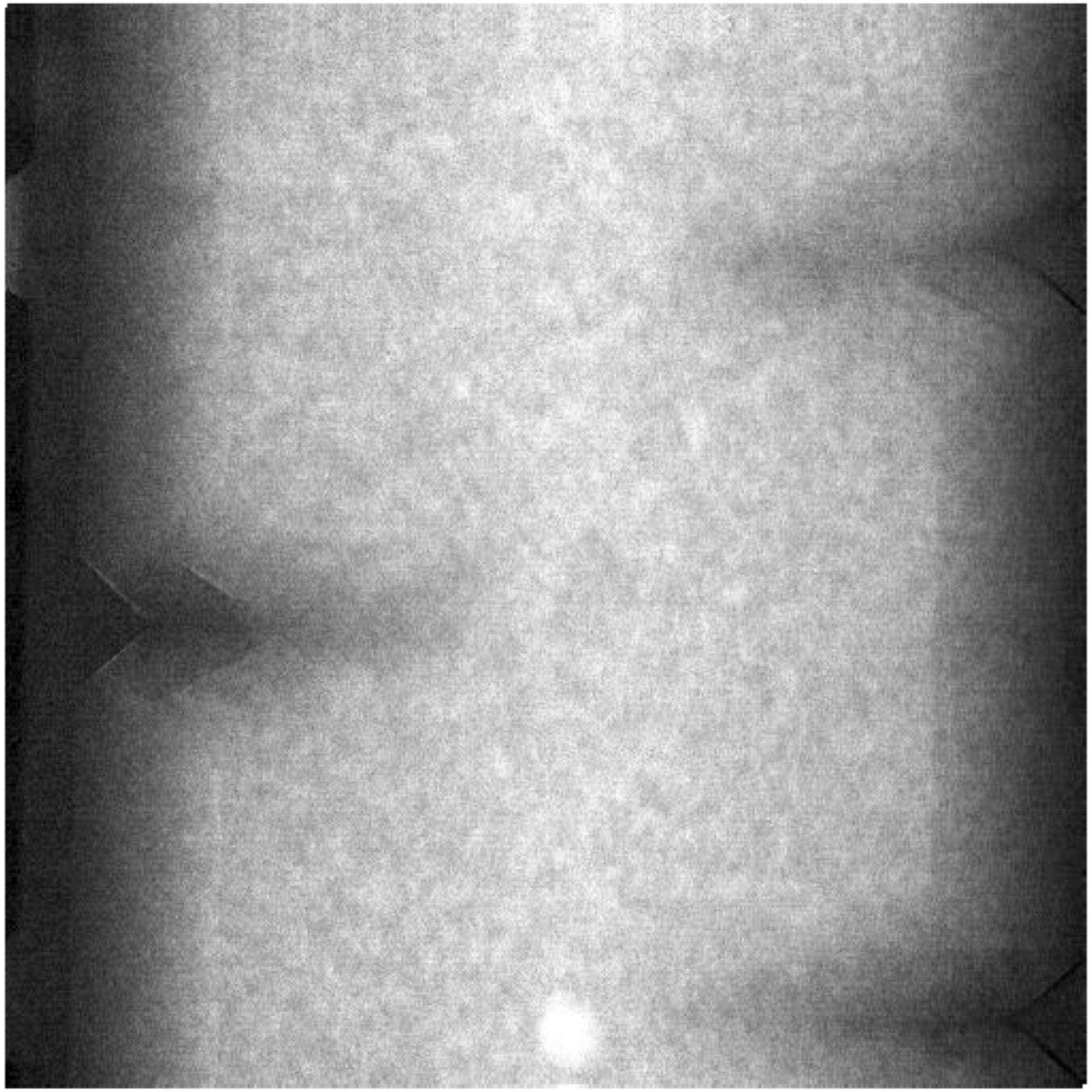
Assume flow constancy over window

- Whose solution is

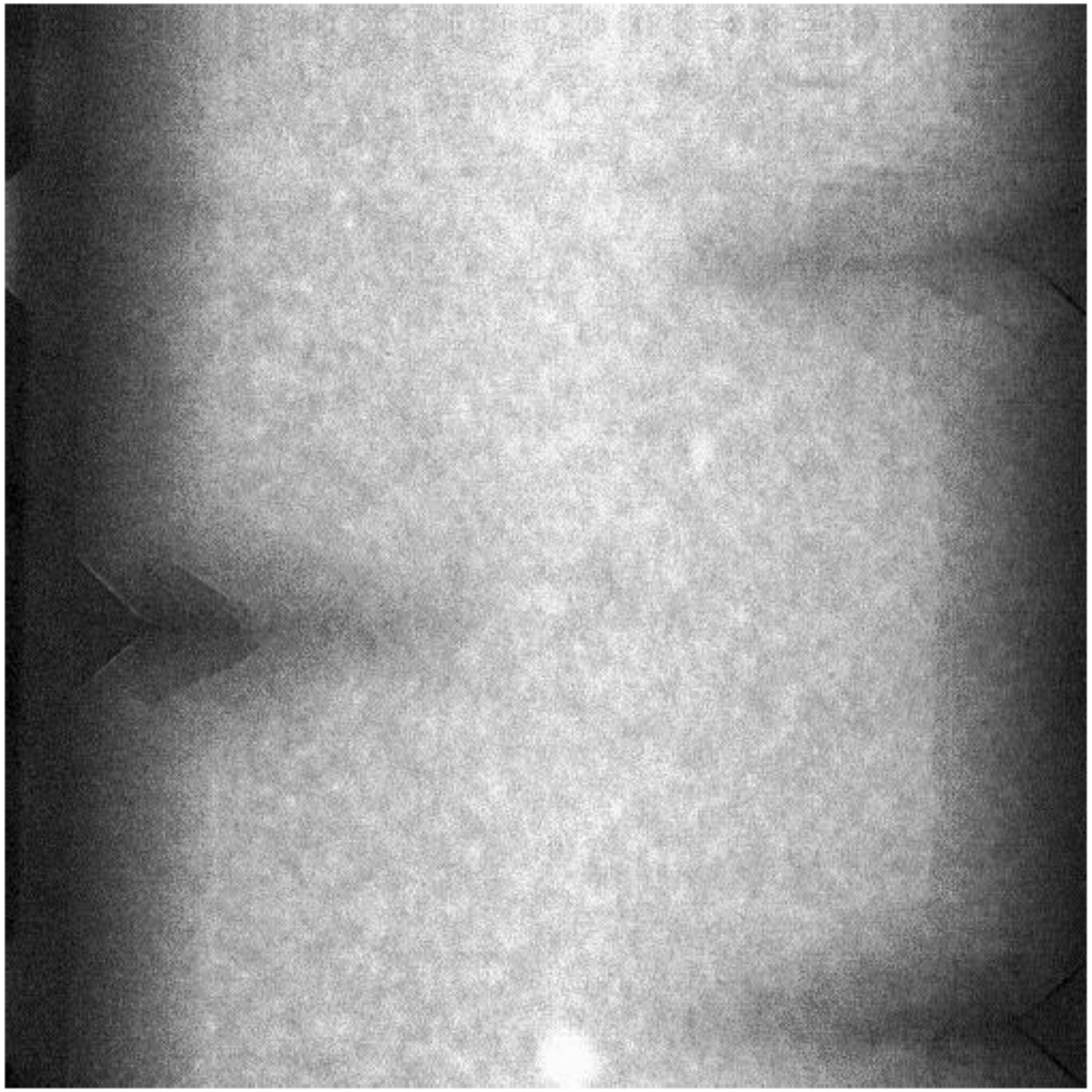
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

  – Window size
    - Smaller windows give better precision
    - Larger windows provide better performance in presence of noise
    - What about a Pyramid?

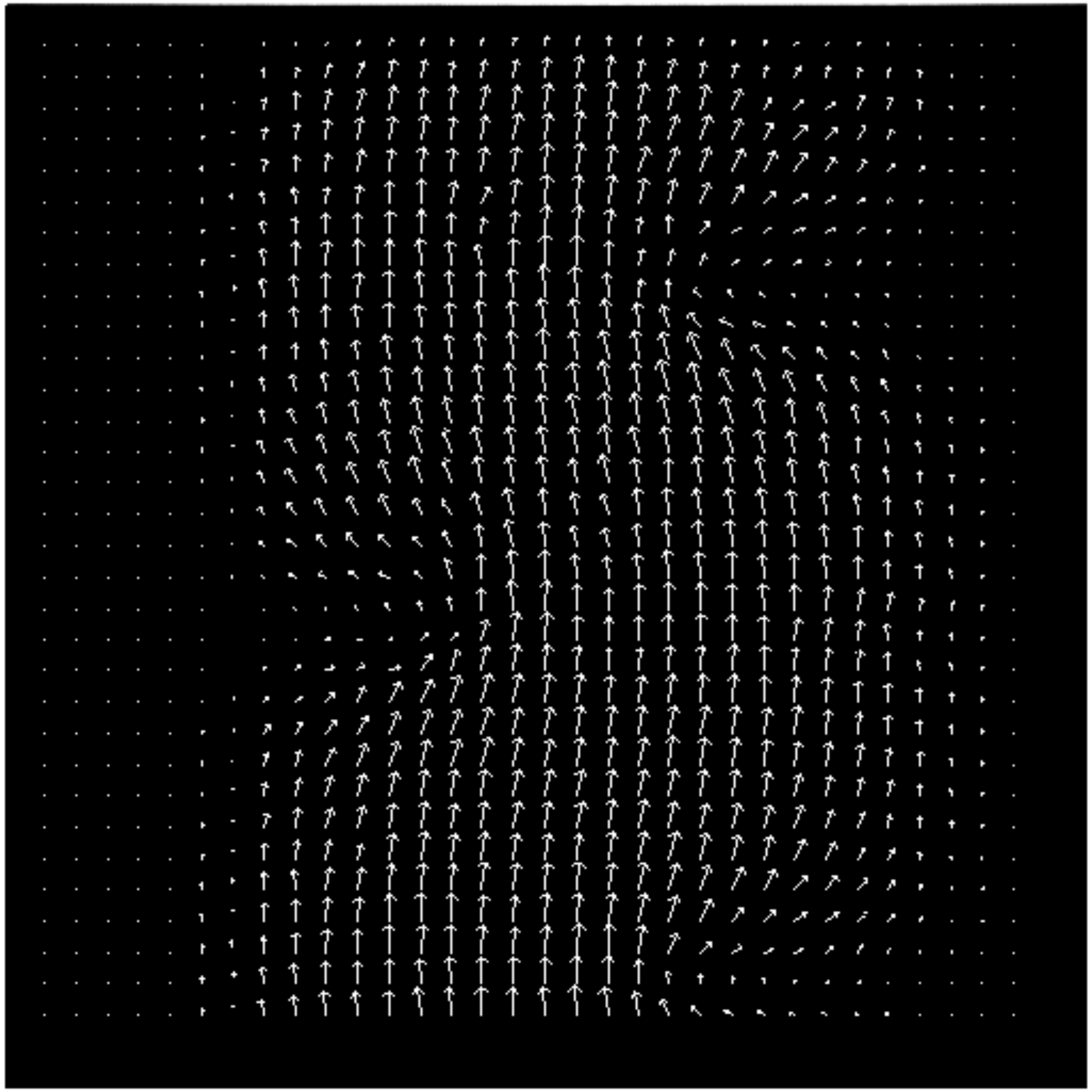
Example



Example



Example



Examples

- [Lukas and Kanade]
- [Horn and Schunk]

Outline

- Introduction
- Motion field vs. optical flow
- Brightness constancy
- Gradient-based optical flow estimation
- Finite displacement and feature-based methods
- Object and scene deformations

Finite displacement and features

- Gradient-based techniques work best when the displacements between the image are relatively small
  - This is implicit in the derivation of the optical flow constraint equation via differentials
  - Although course-to-fine processing can help with this limitation
- Well detected and localized features have the potential to be reasonably matched between images.
- Therefore, such approaches have received attention in conjunction with larger motion displacements.

Finite displacement and features

- **Two broad classes of approach**
- Methods for matching between binocular stereo pairs can be adapted to finite displacement image motion.
  - For example, the feature-based methods are particularly applicable
- Also of interest is the iteration of gradient-based optical flow
  - But restricted to interesting feature points

Finite displacement and features

- Iterated gradient
  - We begin by extracting feature points of interest in image 1 of the input pair.
  - For example, the corner/line detector shown in previous lectures is well suited for this purpose
  - We then center windows about a feature of interest and about the same location in the other image.

Finite displacement and features

- Iterated gradient
  - We begin by extracting feature points of interest in image 1 of the input pair.
  - For example, the corner/line detector shown in previous lectures is well suited for this purpose
  - We then center windows about a feature of interest and about the same location in the other image.
  - We execute the gradient-based calculation, i.e.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

Finite displacement and features

- Iterated gradient
  - We execute the gradient-based calculation, i.e.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

  - Following completion, we shift the entire window about the feature in image 1 according to the recovered flow vector.

Finite displacement and features

- Iterated gradient
  - We execute the gradient-based calculation, i.e.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

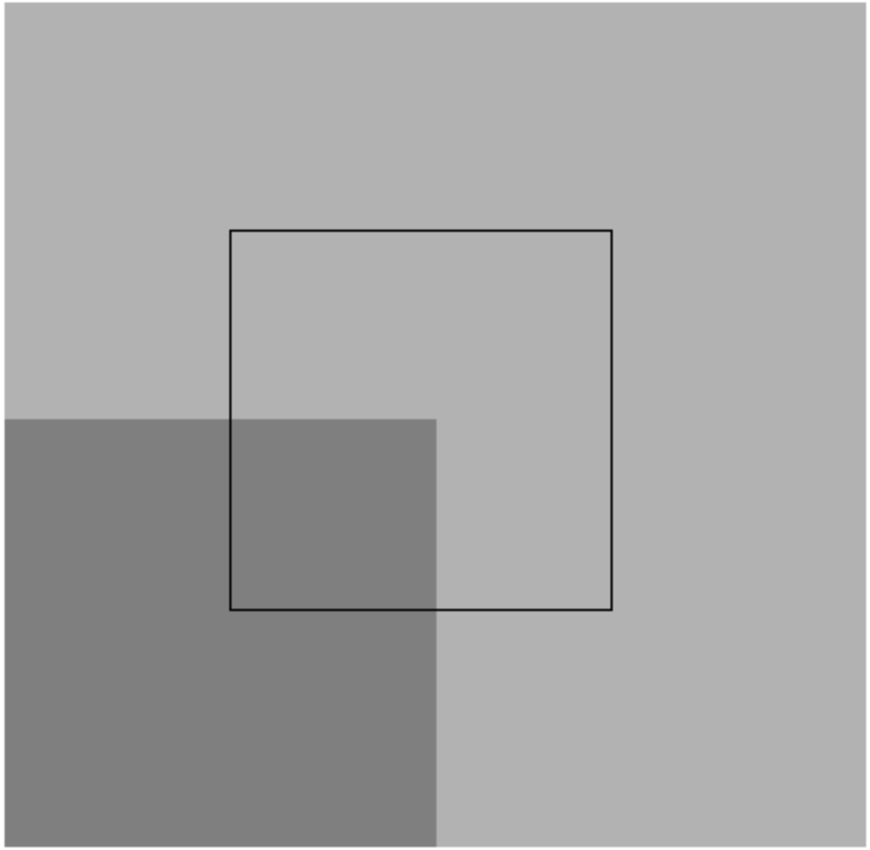
  - Following completion, we shift the entire window about the feature in image 1 according to the recovered flow vector.
  - We then calculate a similarity measure between the shifted window in image 1 and the window in image 2.

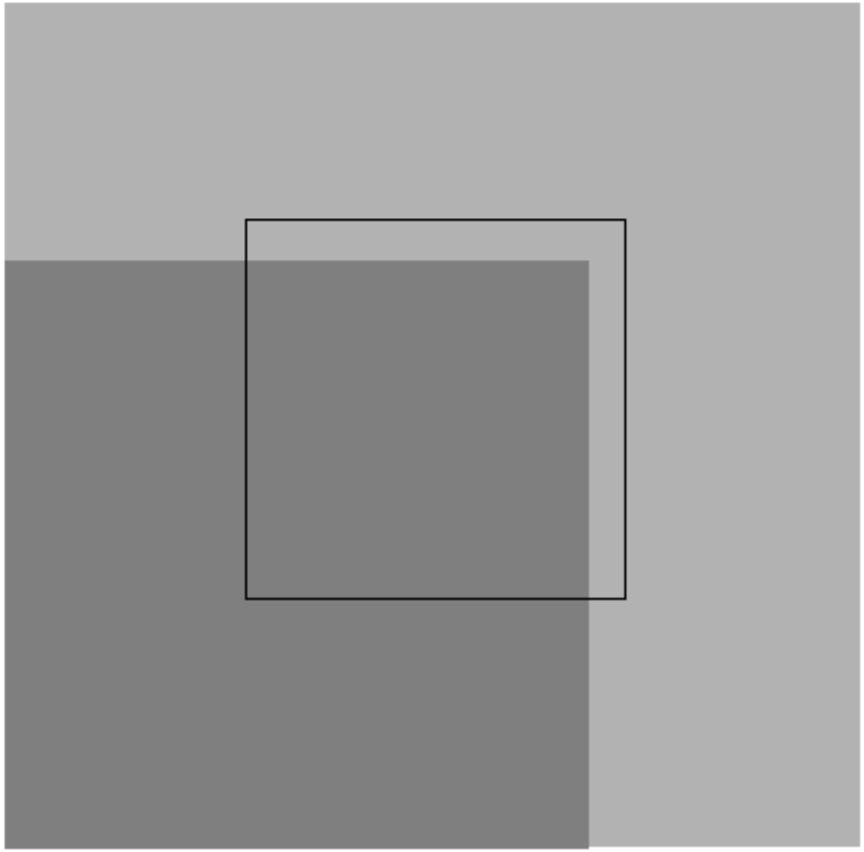
Finite displacement and features

- Iterated gradient
  - Following completion, we shift the entire window about the feature in image 1 according to the recovered flow vector.
  - We then calculate a similarity measure between the shifted window in image 1 and the window in image 2.
  - If the similarity is above some threshold, then we say that the match has been found and we exit.
  - If the similarity measure is below some threshold, then we iterate the gradient-based calculation, but now making use of the shifted window in image 1.

Finite displacement and features

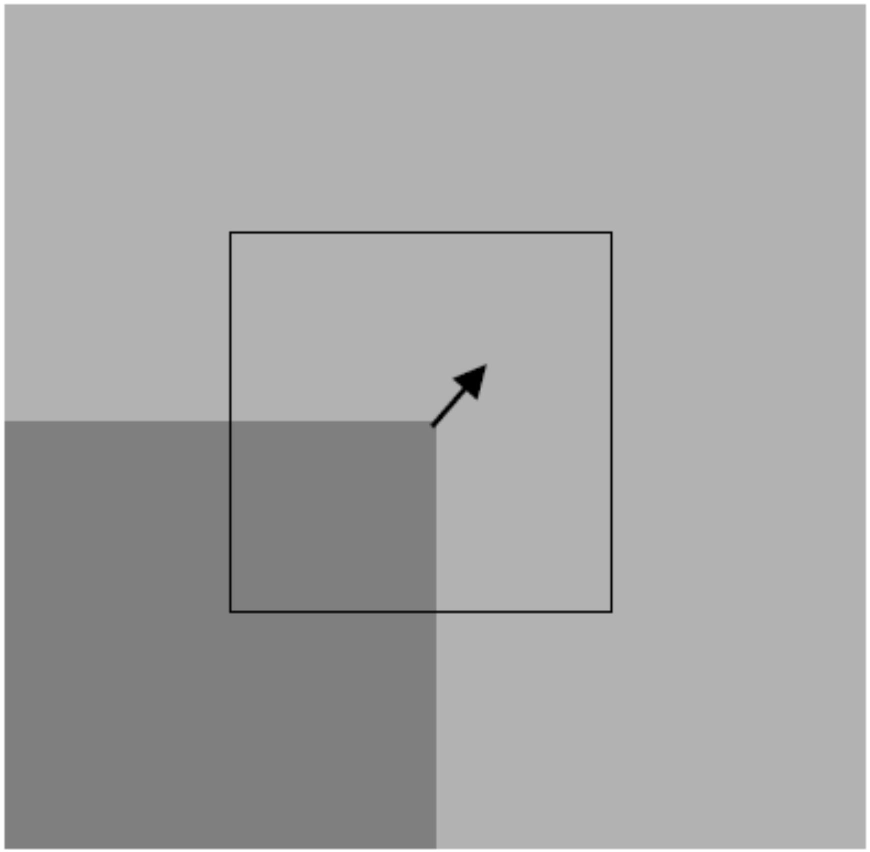
- Iterated gradient

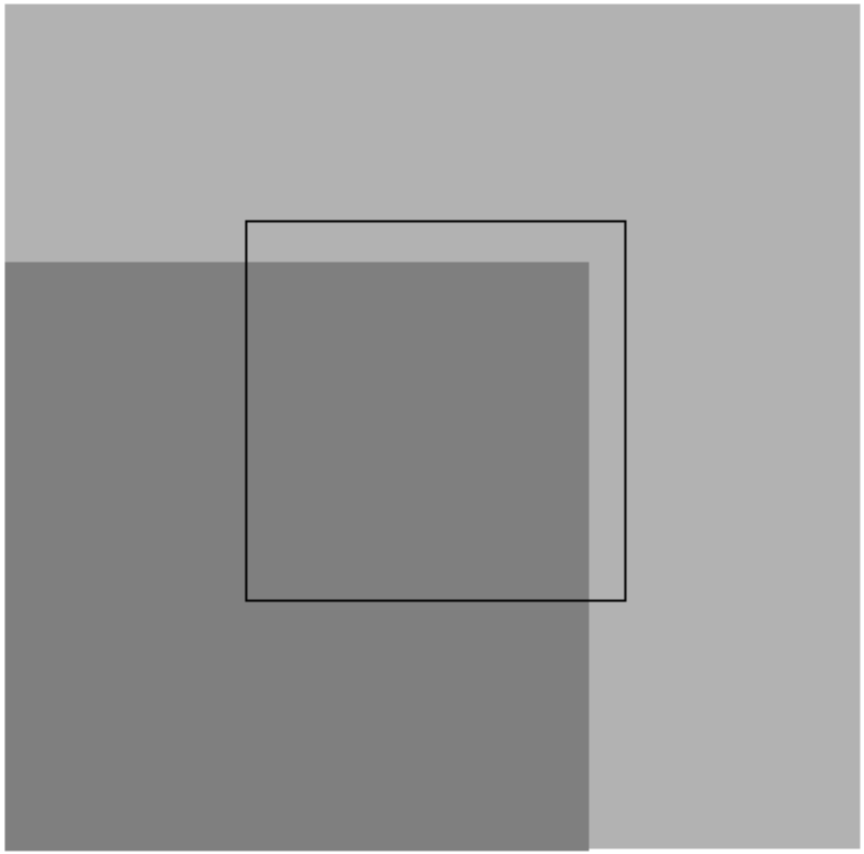




Finite displacement and features

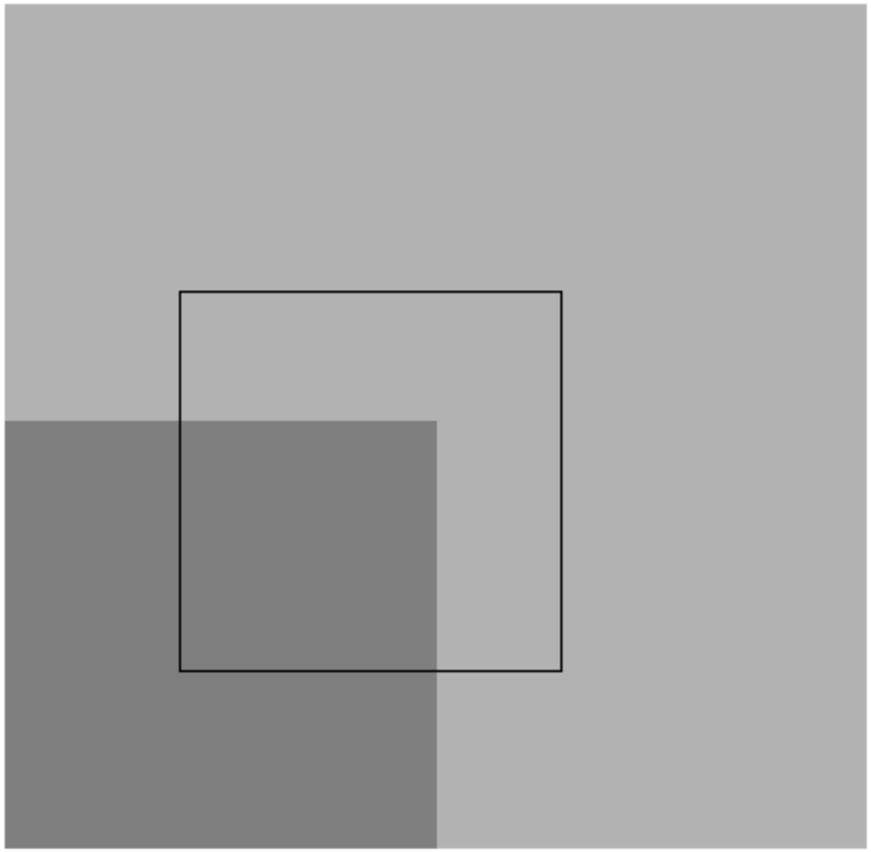
- Iterated gradient

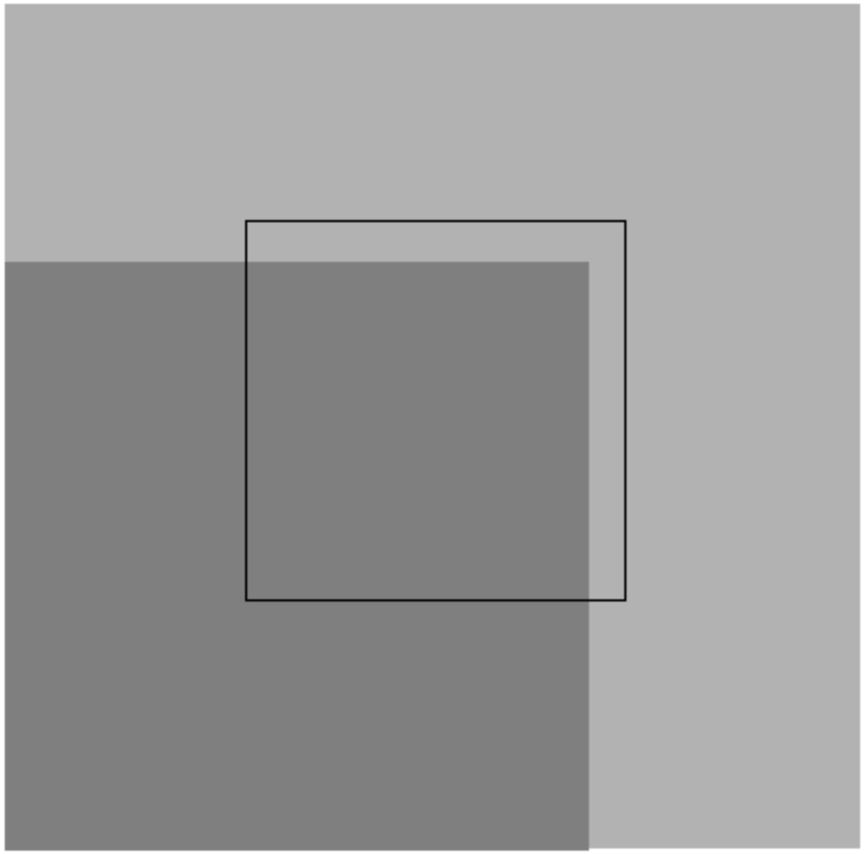




Finite displacement and features

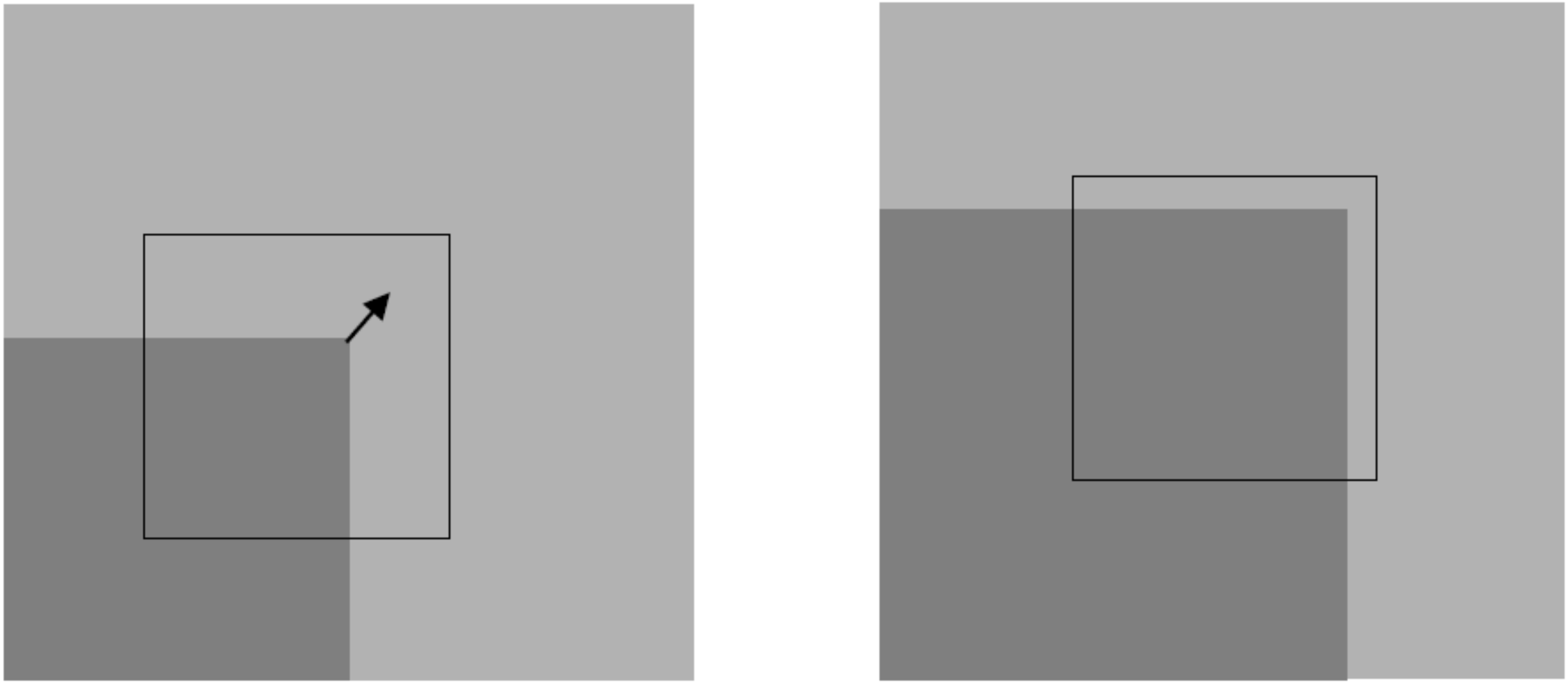
- Iterated gradient





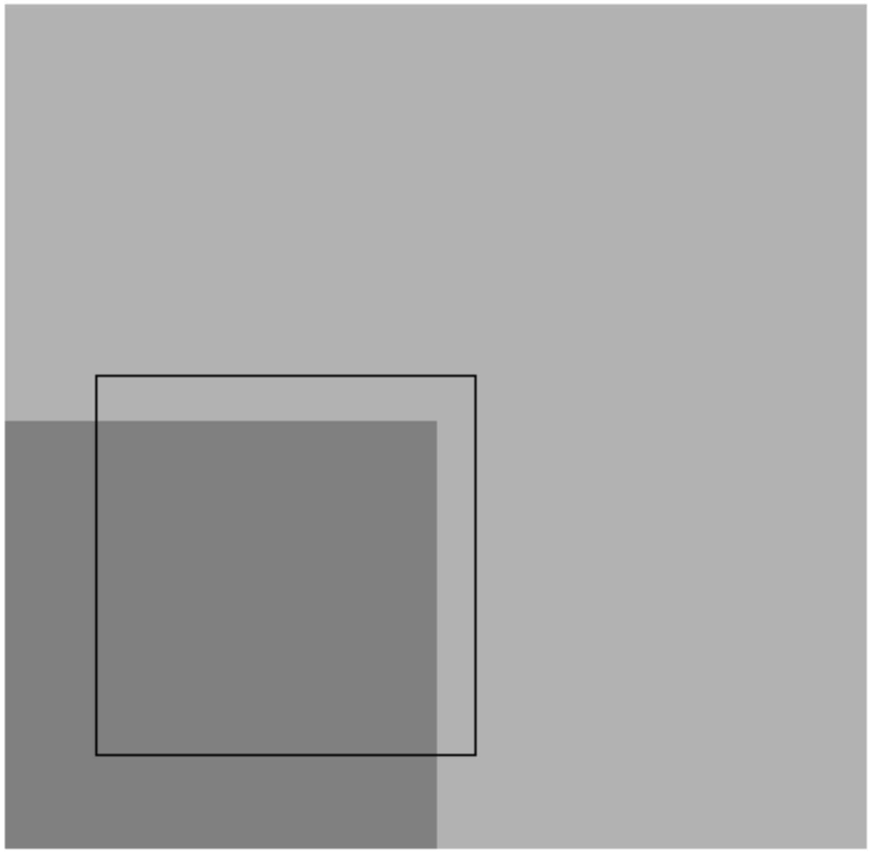
Finite displacement and features

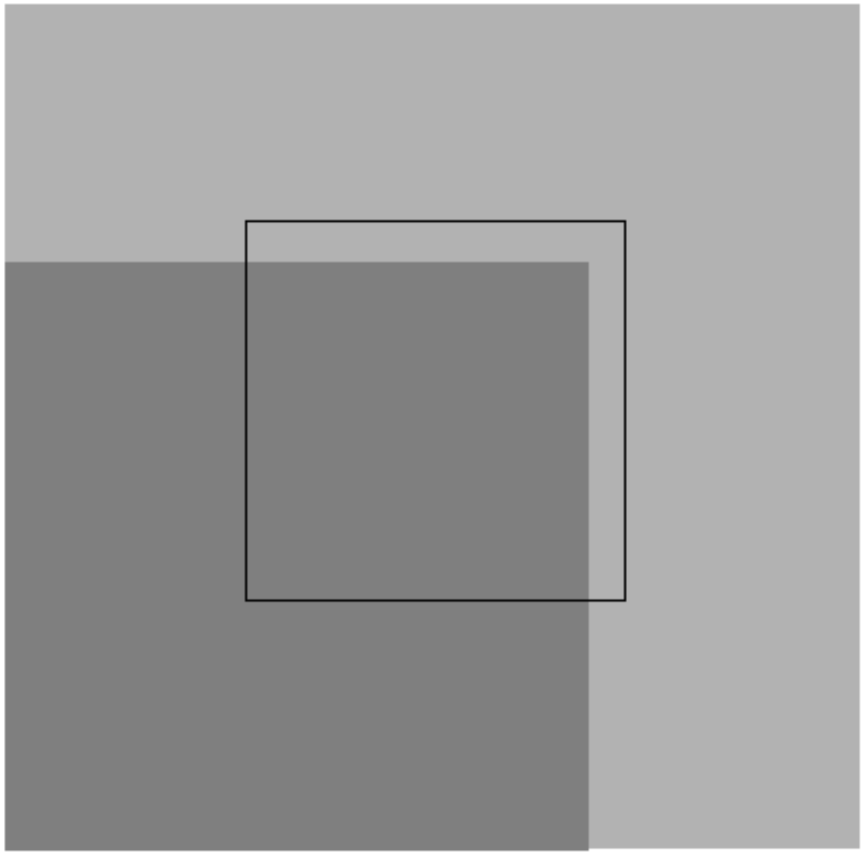
- Iterated gradient



Finite displacement and features

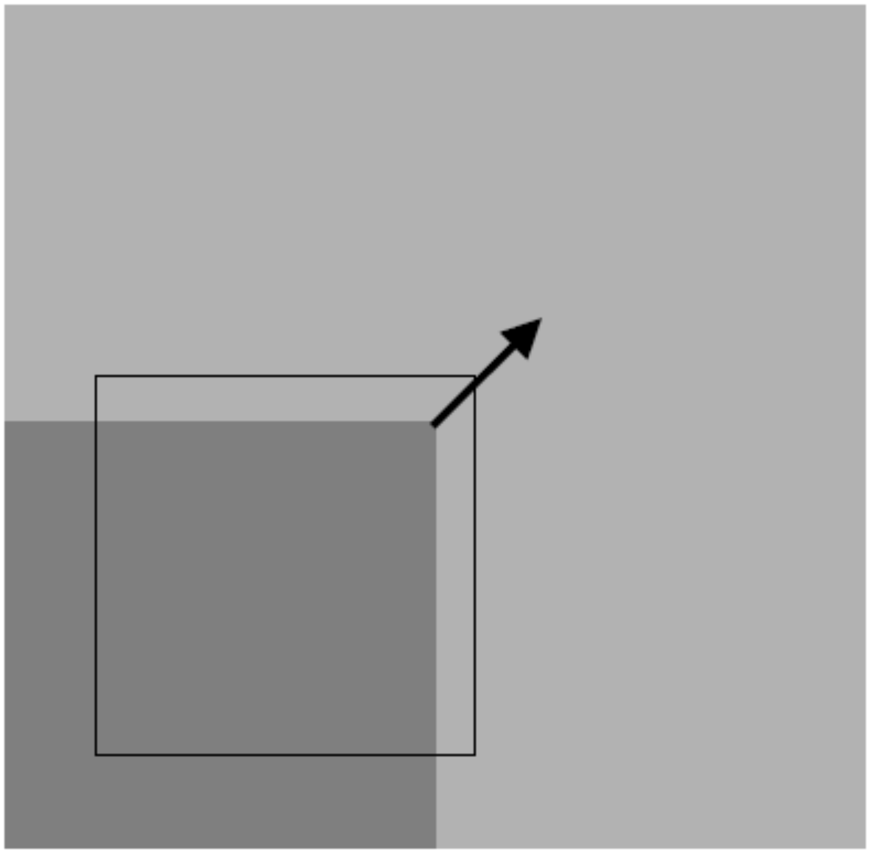
- Iterated gradient

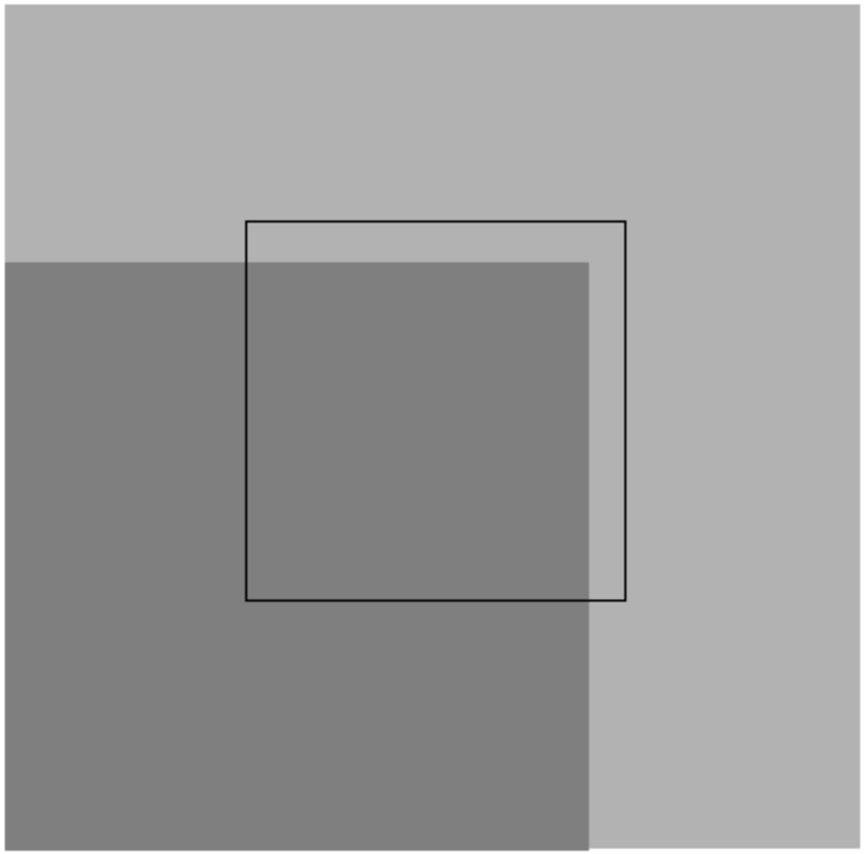




Finite displacement and features

- Iterated gradient





Finite displacement and features

Algorithm

- **Input:** Two images I1 and I2 and a set of features for I1
- **Output:** A set of displacements, one for each feature of I1.
- **Notation:** Let
  - $Q1$, $Q2$ *and* be two image windows
  - t be a threshold, a fixed positive real number
  - $\boldsymbol{p}$ be a feature point in I1
  - $\boldsymbol{d}$ be the unknown displacement for $\boldsymbol{p}$
- For each feature point $\boldsymbol{p}$

1. Set $\boldsymbol{d}$ = 0 and centre $Q1$ on $\boldsymbol{p}$
2. Estimate the displacement $\boldsymbol{d0}$ of $\boldsymbol{p}$ center of $Q1$ according to the gradient-based algorithm
3. Set $\boldsymbol{d}=\boldsymbol{d}+\boldsymbol{d0}$
4. Let $Q2$ be the image patch obtained by shifting $Q1$ according to $\boldsymbol{d0}$.
   - Calculate the similarity, S, of $Q2$ and the corresponding patch in I2
5. If $S<t$ then set $Q1=Q2$ and goto 2; else exit.

Representative similarity measure

- 1/(Sum of Squared Differences) within the windows of interest is a reasonable choice for this algorithm.

Outline

- Introduction
- Motion field vs. optical flow
- Brightness constancy
- Gradient-based optical flow estimation
- Finite displacement and feature-based methods
- Object and scene deformations

Object and scene deformations

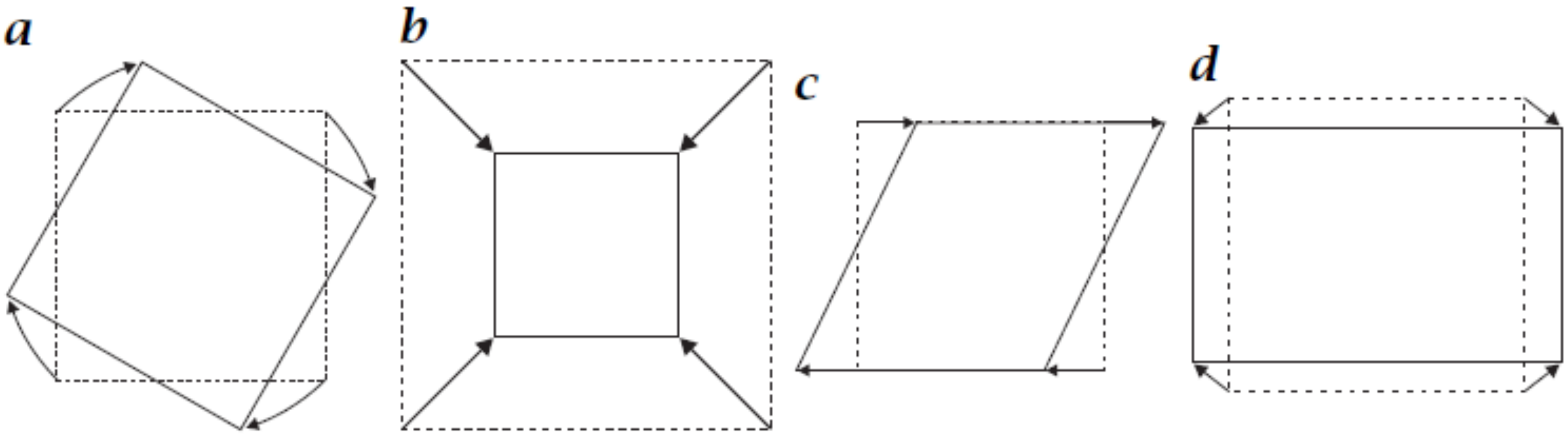


Figure 10.10: *Elementary geometric transformations of a planar surface element undergoing affine transformation:* ***a*** *rotation;* ***b*** *dilation;* ***c*** *shear;* ***d*** *stretching.*

Source: Jähne, Computer Vision and applications

Object and scene deformations

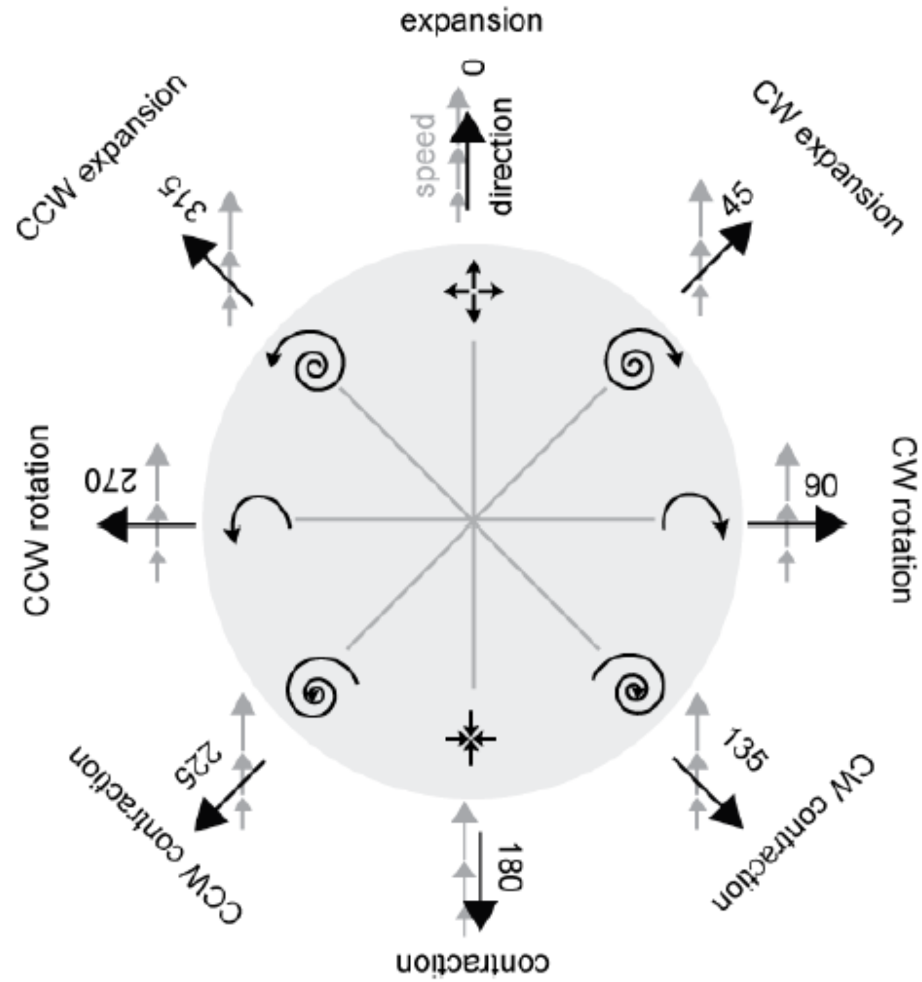


Figure 4.1: Spiral space is a coordinate system that interprets expansion (0°), contraction (180°) and rotations (clockwise: 90°, counterclockwise: 270°) as cardinal directions with in- and outward spiraling movement patterns placed in between. Gray arrows show local motion direction.

Source: Rodriguez-Sanchez et al., 2009

Summary

- Introduction
- Motion field vs. optical flow
- Brightness constancy
- Gradient-based optical flow estimation
- Finite displacement and feature-based methods
- Object and scene deformations

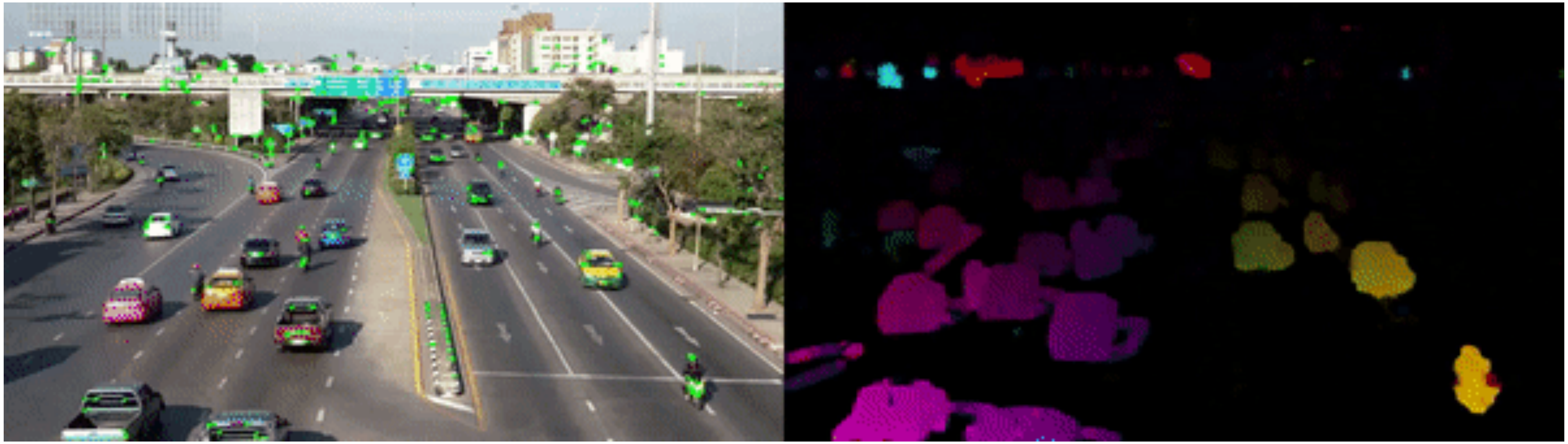
universität innsbruck

Optical flow with OpenCV

Assoz.Prof. Antonio Rodríguez-Sánchez, PhD.

Optical Flow

- Lukas and Kanade
- Dense Optical flow



Assume flow constancy over window

- Lucas and Kanade (1981)
- Flow constancy assumes that over some window, W, the values of (u,v) are constant.

Assume flow constancy over window

- Lucas and Kanade (1981)
- Flow constancy assumes that over some window, ***W***, the values of ***(u,v)*** are constant.
- We seek to minimize

$$\min_{(u,v)} \sum \sum_{W} (E_x u + E_y v + E_t)^2$$

Assume flow constancy over window

- Lucas and Kanade (1981)
- Flow constancy assumes that over some window, ***W***, the values of ***(u,v)*** are constant.
- We seek to minimize

$$\min_{(u,v)} \sum \sum_{W} (E_x u + E_y v + E_t)^2$$

- We solve this by differentiation and equalizing to 0

$$\sum \sum (E_x u + E_y v + E_t) E_x = 0$$

$$\sum \sum (E_x u + E_y v + E_t) E_y = 0$$

Assume flow constancy over window

- Lucas and Kanade (1981)
- Flow constancy assumes that over some window, ***W***, the values of ***(u,v)*** are constant.
- We seek to minimize

$$\min_{(u,v)} \sum \sum_{W} (E_x u + E_y v + E_t)^2$$

- We solve this by differentiation and equalizing to 0

$$\sum \sum (E_x u + E_y v + E_t) E_x = 0$$
$$\sum \sum (E_x u + E_y v + E_t) E_y = 0$$

$\Rightarrow$ **Two equations with two unknowns !**

Assume flow constancy over window

$$\sum\sum(E_x u + E_y v + E_t)E_x = 0$$
$$\sum\sum(E_x u + E_y v + E_t)E_y = 0$$

Two equations with two unknowns !

- Whose solution is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

Assume flow constancy over window

- Whose solution is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

  – Input: A temporal sequence of two images
  – Output: A pair of optical flow images, a U an V image
  – We compute the equation above and store the recovered ***(u,v)*** in the positions ***(i,j)*** in the U and V images

Assume flow constancy over window

- Whose solution is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum\sum E_x^2 & \sum\sum E_x E_y \\ \sum\sum E_x E_y & \sum\sum E_y^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum\sum E_x E_t \\ -\sum\sum E_y E_t \end{pmatrix}$$

  - Window size
    - Smaller windows give better precision
    - Larger windows provide better performance in presence of noise

Lukas and Kanade

```
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/video.hpp>

using namespace cv;
using namespace std;

int main(int argc, char **argv)
{
    const string about =
        "This sample demonstrates Lucas-Kanade Optical Flow calculation.\n"
        "The example file can be downloaded from:\n"
        "  https://www.bogotobogo.com/python/OpenCV_Python/images/mean_shift_tracking/slow_traffic_small.mp4";
    const string keys =
        "{ h help |      | print this help message }"
        "{ @image | vtest.avi | path to image file }";
    CommandLineParser parser(argc, argv, keys);
    parser.about(about);
    if (parser.has("help"))
    {
        parser.printMessage();
        return 0;
    }
    string filename = samples::findFile(parser.get<string>("@image"));
    if (!parser.check())
    {
        parser.printErrors();
        return 0;
    }

    VideoCapture capture(filename);
    if (!capture.isOpened()){
        //error in opening the video input
        cerr << "Unable to open file!" << endl;
        return 0;
    }
```

Lukas and Kanade

```
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/video.hpp>

using namespace cv;
using namespace std;

int main(int argc, char **argv)
{
    const string about =
        "This sample demonstrates Lucas-Kanade Optical Flow calculation.\n"
        "The example file can be downloaded from:\n"
        "  https://www.bogotobogo.com/python/OpenCV_Python/images/mean_shift_tracking/slow_traffic_small.mp4";
    const string keys =
        "{ h help |      | print this help message }"
        "{ @image | vtest.avi | path to image file }";
    CommandLineParser parser(argc, argv, keys);
    parser.about(about);
    if (parser.has("help"))
    {
        parser.printMessage();
        return 0;
    }
    string filename = samples::findFile(parser.get<string>("@image"));
    if (!parser.check())
    {
        parser.printErrors();
        return 0;
    }

    VideoCapture capture(filename);
    if (!capture.isOpened()){
        //error in opening the video input
        cerr << "Unable to open file!" << endl;
        return 0;
    }
```

Lukas and Kanade

```
// Create some random colors
vector<Scalar> colors;
RNG rng;
for(int i = 0; i < 100; i++)
{
    int r = rng.uniform(0, 256);
    int g = rng.uniform(0, 256);
    int b = rng.uniform(0, 256);
    colors.push_back(Scalar(r,g,b));
}

Mat old_frame, old_gray;
vector<Point2f> p0, p1;

// Take first frame and find corners in it
capture >> old_frame;
cvtColor(old_frame, old_gray, COLOR_BGR2GRAY);
goodFeaturesToTrack(old_gray, p0, 100, 0.3, 7, Mat(), 7, false, 0.04);

// Create a mask image for drawing purposes
Mat mask = Mat::zeros(old_frame.size(), old_frame.type());

while(true){
    Mat frame, frame_gray;

    capture >> frame;
    if (frame.empty())
        break;
    cvtColor(frame, frame_gray, COLOR_BGR2GRAY);
```

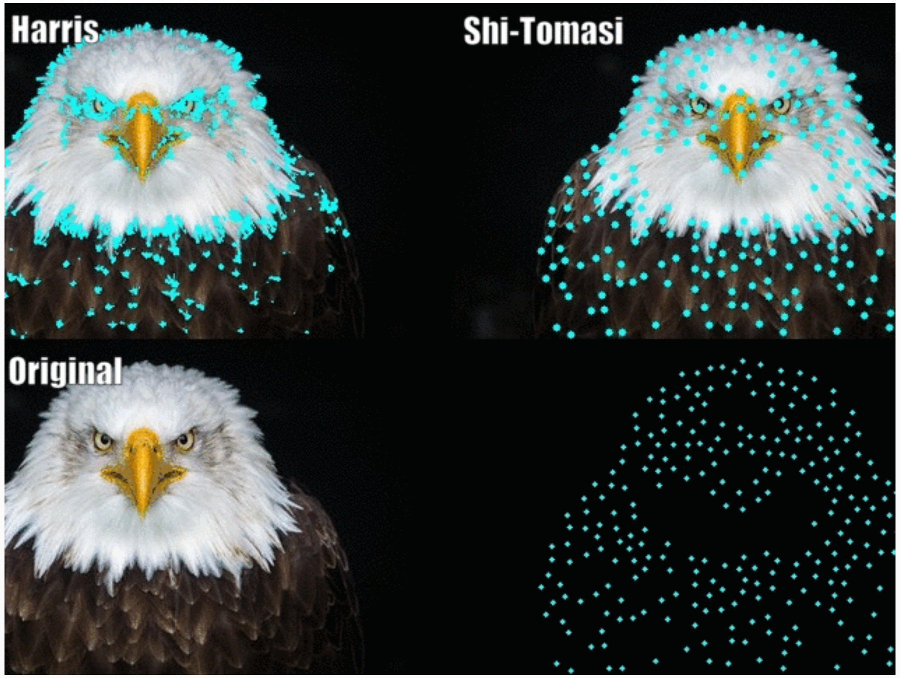
Lukas and Kanade

```
// Create some random colors
vector<Scalar> colors;
RNG rng;
for(int i = 0; i < 100; i++)
{
    int r = rng.uniform(0, 256);
    int g = rng.uniform(0, 256);
    int b = rng.uniform(0, 256);
    colors.push_back(Scalar(r,g,b));
}

Mat old_frame, old_gray;
vector<Point2f> p0, p1;

// Take first frame and find corners in it
capture >> old_frame;
cvtColor(old_frame, old_gray, COLOR_BGR2GRAY);
goodFeaturesToTrack(old_gray, p0, 100, 0.3, 7, Mat(), 7, false, 0.04);

// Create a mask image for drawing purposes
Mat mask = Mat::zeros(old_frame.size(), old_frame.type());

while(true){
    Mat frame, frame_gray;

    capture >> frame;
    if (frame.empty())
        break;
    cvtColor(frame, frame_gray, COLOR_BGR2GRAY);
```

Lukas and Kanade

```
// Create some random colors
vector<Scalar> colors;
RNG rng;
for(int i = 0; i < 100; i++)
{
    int r = rng.uniform(0, 256);
    int g = rng.uniform(0, 256);
    int b = rng.uniform(0, 256);
    colors.push_back(Scalar(r,g,b));
}

Mat old_frame, old_gray;
vector<Point2f> p0, p1;

// Take first frame and find corners in it
capture >> old_frame;
cvtColor(old_frame, old_gray, COLOR_BGR2GRAY);
goodFeaturesToTrack(old_gray, p0, 100, 0.3, 7, Mat(), 7, false, 0.04);

// Create a mask image for drawing purposes
Mat mask = Mat::zeros(old_frame.size(), old_frame.type());

while(true){
    Mat frame, frame_gray;

    capture >> frame;
    if (frame.empty())
        break;
    cvtColor(frame, frame_gray, COLOR_BGR2GRAY);
```

Lukas and Kanade




```
// Create some random colors
vector<Scalar> colors;
RNG rng;
for(int i = 0; i < 100; i++)
{
    int r = rng.uniform(0, 256);
    int g = rng.uniform(0, 256);
    int b = rng.uniform(0, 256);
    colors.push_back(Scalar(r,g,b));
}

Mat old_frame, old_gray;
vector<Point2f> p0, p1;

// Take first frame and find corners in it
capture >> old_frame;
cvtColor(old_frame, old_gray, COLOR_BGR2GRAY);
goodFeaturesToTrack(old_gray, p0, 100, 0.3, 7, Mat(), 7, false, 0.04);

// Create a mask image for drawing purposes
Mat mask = Mat::zeros(old_frame.size(), old_frame.type());

while(true){
    Mat frame, frame_gray;

    capture >> frame;
    if (frame.empty())
        break;
    cvtColor(frame, frame_gray, COLOR_BGR2GRAY);
```

Lukas and Kanade

```
    while(true){
        Mat frame, frame_gray;

        capture >> frame;
        if (frame.empty())
            break;
        cvtColor(frame, frame_gray, COLOR_BGR2GRAY);

        // calculate optical flow
        vector<uchar> status;
        vector<float> err;
        TermCriteria criteria = TermCriteria((TermCriteria::COUNT) + (TermCriteria::EPS), 10, 0.03);
        calcOpticalFlowPyrLK(old_gray, frame_gray, p0, p1, status, err, Size(15,15), 2, criteria);

        vector<Point2f> good_new;
        for(uint i = 0; i < p0.size(); i++)
        {
            // Select good points
            if(status[i] == 1) {
                good_new.push_back(p1[i]);
                // draw the tracks
                line(mask,p1[i], p0[i], colors[i], 2);
                circle(frame, p1[i], 5, colors[i], -1);
            }
        }
        Mat img;
        add(frame, mask, img);

        imshow("Frame", img);

        int keyboard = waitKey(30);
        if (keyboard == 'q' || keyboard == 27)
            break;

        // Now update the previous frame and previous points
        old_gray = frame_gray.clone();
        p0 = good_new;
    }
}
```

Lukas and Kanade

```
        // calculate optical flow
        vector<uchar> status;
        vector<float> err;
        TermCriteria criteria = TermCriteria((TermCriteria::COUNT) + (TermCriteria::EPS), 10, 0.03);
        calcOpticalFlowPyrLK(old_gray, frame_gray, p0, p1, status, err, Size(15,15), 2, criteria);

        vector<Point2f> good_new;
        for(uint i = 0; i < p0.size(); i++)
        {
            // Select good points
            if(status[i] == 1) {
                good_new.push_back(p1[i]);
                // draw the tracks
                line(mask,p1[i], p0[i], colors[i], 2);
                circle(frame, p1[i], 5, colors[i], -1);
            }
        }
        Mat img;
        add(frame, mask, img);

        imshow("Frame", img);

        int keyboard = waitKey(30);
        if (keyboard == 'q' || keyboard == 27)
            break;

        // Now update the previous frame and previous points
        old_gray = frame_gray.clone();
        p0 = good_new;
    }
}
```

Lukas and Kanade

```
        // calculate optical flow
        vector<uchar> status;
        vector<float> err;
        TermCriteria criteria = TermCriteria((TermCriteria::COUNT) + (TermCriteria::EPS), 10, 0.03);
        calcOpticalFlowPyrLK(old_gray, frame_gray, p0, p1, status, err, Size(15,15), 2, criteria);

        vector<Point2f> good_new;
        for(uint i = 0; i < p0.size(); i++)
        {
            // Select good points
            if(status[i] == 1) {
                good_new.push_back(p1[i]);
                // draw the tracks
                line(mask,p1[i], p0[i], colors[i], 2);
                circle(frame, p1[i], 5, colors[i], -1);
            }
        }
        Mat img;
        add(frame, mask, img);

        imshow("Frame", img);

        int keyboard = waitKey(30);
        if (keyboard == 'q' || keyboard == 27)
            break;

        // Now update the previous frame and previous points
        old_gray = frame_gray.clone();
        p0 = good_new;
    }
}
```

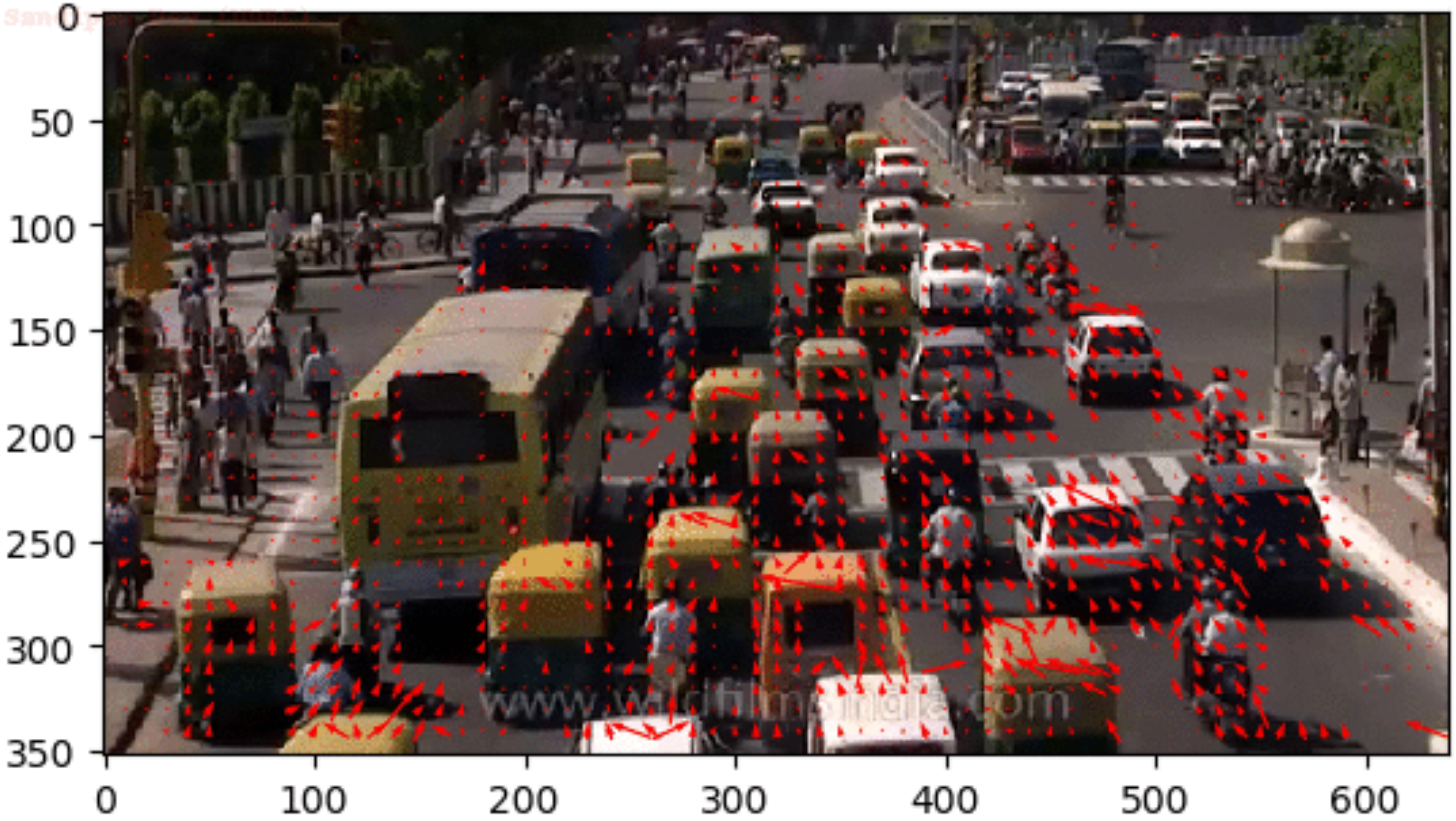
Lukas and Kanade

```
        // calculate optical flow
        vector<uchar> status;
        vector<float> err;
        TermCriteria criteria = TermCriteria((TermCriteria::COUNT) + (TermCriteria::EPS), 10, 0.03);
        calcOpticalFlowPyrLK(old_gray, frame_gray, p0, p1, status, err, Size(15,15), 2, criteria);

        vector<Point2f> good_new;
        for(uint i = 0; i < p0.size(); i++)
        {
            // Select good points
            if(status[i] == 1) {
                good_new.push_back(p1[i]);
                // draw the tracks
                line(mask,p1[i], p0[i], colors[i], 2);
                circle(frame, p1[i], 5, colors[i], -1);
            }
        }
        Mat img;
        add(frame, mask, img);

        imshow("Frame", img);

        int keyboard = waitKey(30);
        if (keyboard == 'q' || keyboard == 27)
            break;

        // Now update the previous frame and previous points
        old_gray = frame_gray.clone();
        p0 = good_new;
    }
}
```

Lukas and Kanade

```
    while(true){
        Mat frame, frame_gray;

        capture >> frame;
        if (frame.empty())
            break;
        cvtColor(frame, frame_gray, COLOR_BGR2GRAY);

        // calculate optical flow
        vector<uchar> status;
        vector<float> err;
        TermCriteria criteria = TermCriteria((TermCriteria::COUNT) + (TermCriteria::EPS), 10, 0.03);
        calcOpticalFlowPyrLK(old_gray, frame_gray, p0, p1, status, err, Size(15,15), 2, criteria);

        vector<Point2f> good_new;
        for(uint i = 0; i < p0.size(); i++)
        {
            // Select good points
            if(status[i] == 1) {
                good_new.push_back(p1[i]);
                // draw the tracks
                line(mask,p1[i], p0[i], colors[i], 2);
                circle(frame, p1[i], 5, colors[i], -1);
            }
        }
        Mat img;
        add(frame, mask, img);

        imshow("Frame", img);

        int keyboard = waitKey(30);
        if (keyboard == 'q' || keyboard == 27)
            break;

        // Now update the previous frame and previous points
        old_gray = frame_gray.clone();
        p0 = good_new;
    }
}
```

Lukas and Kanade



Dense optical flow

```
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/video.hpp>

using namespace cv;
using namespace std;

int main()
{
    VideoCapture capture(samples::findFile("vtest.avi"));
    if (!capture.isOpened()){
        //error in opening the video input
        cerr << "Unable to open file!" << endl;
        return 0;
    }

    Mat frame1, prvs;
    capture >> frame1;
    cvtColor(frame1, prvs, COLOR_BGR2GRAY);
```

Dense optical flow

```
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/video.hpp>

using namespace cv;
using namespace std;

int main()
{
    VideoCapture capture(samples::findFile("vtest.avi"));
    if (!capture.isOpened()){
        //error in opening the video input
        cerr << "Unable to open file!" << endl;
        return 0;
    }

    Mat frame1, prvs;
    capture >> frame1;
    cvtColor(frame1, prvs, COLOR_BGR2GRAY);
```

Dense optical flow

```
while(true){
    Mat frame2, next;
    capture >> frame2;
    if (frame2.empty())
        break;
    cvtColor(frame2, next, COLOR_BGR2GRAY);
```

Dense optical flow

```
while(true){
    Mat frame2, next;
    capture >> frame2;
    if (frame2.empty())
        break;
    cvtColor(frame2, next, COLOR_BGR2GRAY);

    Mat flow(prvs.size(), CV_32FC2);
    calcOpticalFlowFarneback(prvs, next, flow, 0.5, 3, 15, 3, 5, 1.2, 0);
```

Dense optical flow

```
while(true){
    Mat frame2, next;
    capture >> frame2;
    if (frame2.empty())
        break;
    cvtColor(frame2, next, COLOR_BGR2GRAY);

    Mat flow(prvs.size(), CV_32FC2);
    calcOpticalFlowFarneback(prvs, next, flow, 0.5, 3, 15, 3, 5, 1.2, 0);

    // visualization
    Mat flow_parts[2];
    split(flow, flow_parts);
    Mat magnitude, angle, magn_norm;
    cartToPolar(flow_parts[0], flow_parts[1], magnitude, angle, true);
    normalize(magnitude, magn_norm, 0.0f, 1.0f, NORM_MINMAX);
    angle *= ((1.f / 360.f) * (180.f / 255.f));
```

Dense optical flow

```
while(true){
    Mat frame2, next;
    capture >> frame2;
    if (frame2.empty())
        break;
    cvtColor(frame2, next, COLOR_BGR2GRAY);

    Mat flow(prvs.size(), CV_32FC2);
    calcOpticalFlowFarneback(prvs, next, flow, 0.5, 3, 15, 3, 5, 1.2, 0);

    // visualization
    Mat flow_parts[2];
    split(flow, flow_parts);
    Mat magnitude, angle, magn_norm;
    cartToPolar(flow_parts[0], flow_parts[1], magnitude, angle, true);
    normalize(magnitude, magn_norm, 0.0f, 1.0f, NORM_MINMAX);
    angle *= ((1.f / 360.f) * (180.f / 255.f));

    //build hsv image
    Mat _hsv[3], hsv, hsv8, bgr;
    _hsv[0] = angle;
    _hsv[1] = Mat::ones(angle.size(), CV_32F);
    _hsv[2] = magn_norm;
    merge(_hsv, 3, hsv);
    hsv.convertTo(hsv8, CV_8U, 255.0);
    cvtColor(hsv8, bgr, COLOR_HSV2BGR);

    imshow("frame2", bgr);

    int keyboard = waitKey(30);
    if (keyboard == 'q' || keyboard == 27)
        break;

    prvs = next;
}
```

Dense optical flow

Optical flow

- Tutorial: https://nanonets.com/blog/optical-flow/
- Code: https://docs.opencv.org/3.4/dc/d6b/group__video__track.html#ga5d10ebbd59fe09c5f650289ec0ece5af
- Faneback's paper: http://www.diva-portal.org/smash/get/diva2:273847/FULLTEXT01.pdf