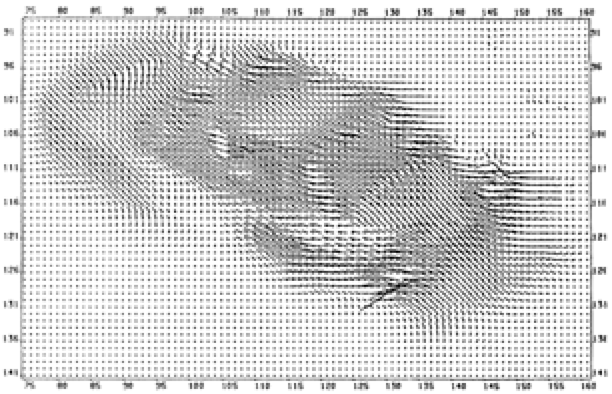
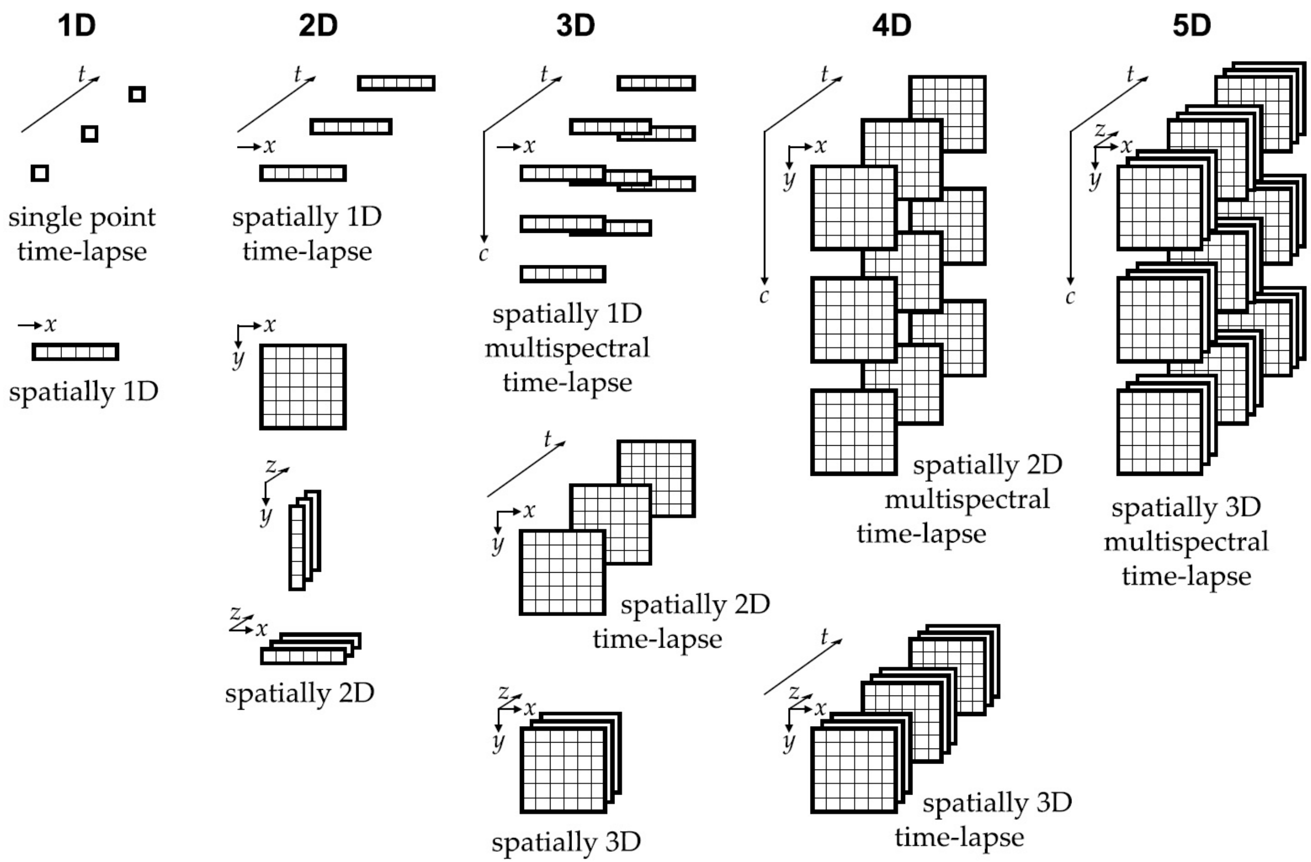


COMP9517: Computer Vision

Motion

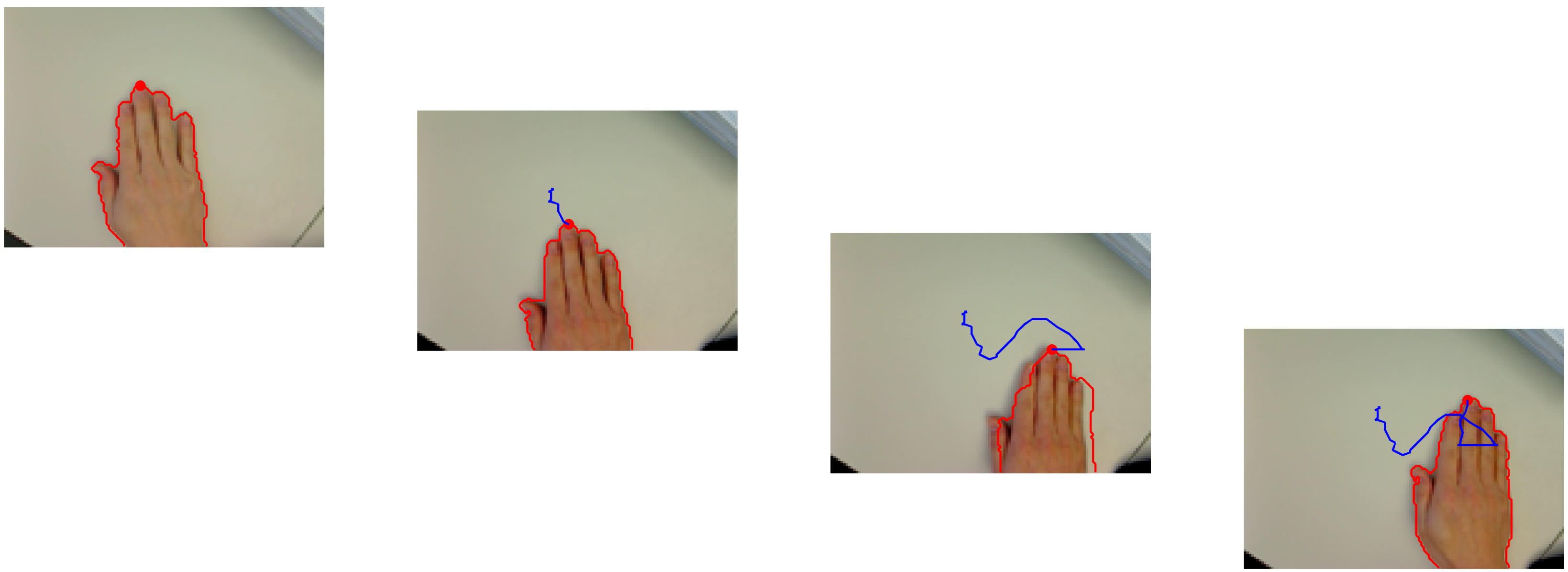
Introduction

- Adding the time dimension to the image formation



Introduction

- A changing scene may be observed via a sequence of images



Introduction

- Changes in an image sequence provide features for
  - detecting objects that are moving
  - computing trajectories of moving objects
  - performing motion analysis of moving objects
  - recognising objects based on their behaviours
  - computing the motion of the viewer in the world
  - detecting and recognising activities in a scene

Applications

- **Motion-based recognition**
  – human identification based on gait, automatic object detection
- **Automated surveillance**
  – monitoring a scene to detect suspicious activities or unlikely events
- **Video indexing**
  – automatic annotation and retrieval of videos in multimedia databases
- **Human-computer interaction**
  – gesture recognition, eye gaze tracking for data input to computers
- **Traffic monitoring**
  – real-time gathering of traffic statistics to direct traffic flow
- **Vehicle navigation**
  – video-based path planning and obstacle avoidance capabilities

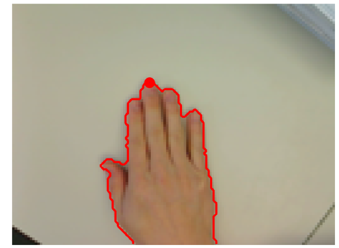
Scenarios

- Still camera

  Constant background with

  – single moving object

  – multiple moving objects



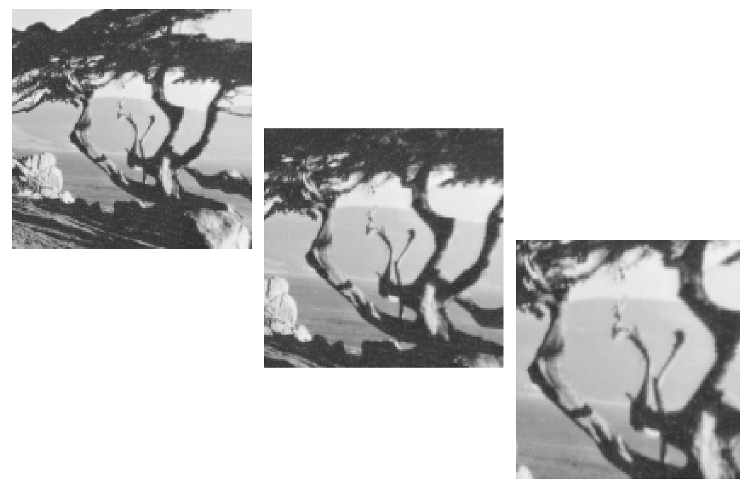
- Moving camera

  Relatively constant scene with

  – coherent scene motion

  – single moving object

  – multiple moving objects



Topics

- **Change detection**

  Using *image subtraction* to detect changes in scenes

- **Sparse motion estimation**

  Using *template matching* to estimate local displacements

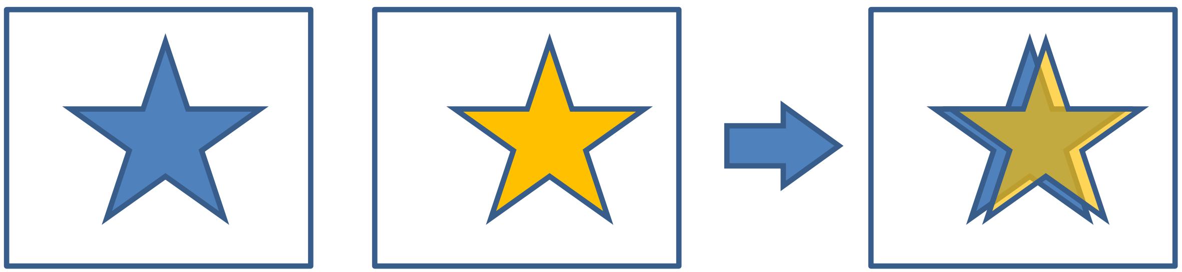
- **Dense motion estimation**

  Using *optical flow* to compute a dense motion vector field

Change Detection

Change Detection

- Detecting an object moving across a constant background
- The forward and rear edges of the object advance only a few pixels per frame



- By subtracting the image I_t from the previous image I_{t-1} the edges should be evident as the only pixels significantly different from zero

Image Subtraction

Step: Derive a background image from a set of video frames at the beginning of the video sequence

PETS 2009 Benchmark

Image Subtraction

Step: Subtract the background image from each subsequent frame to create a difference image

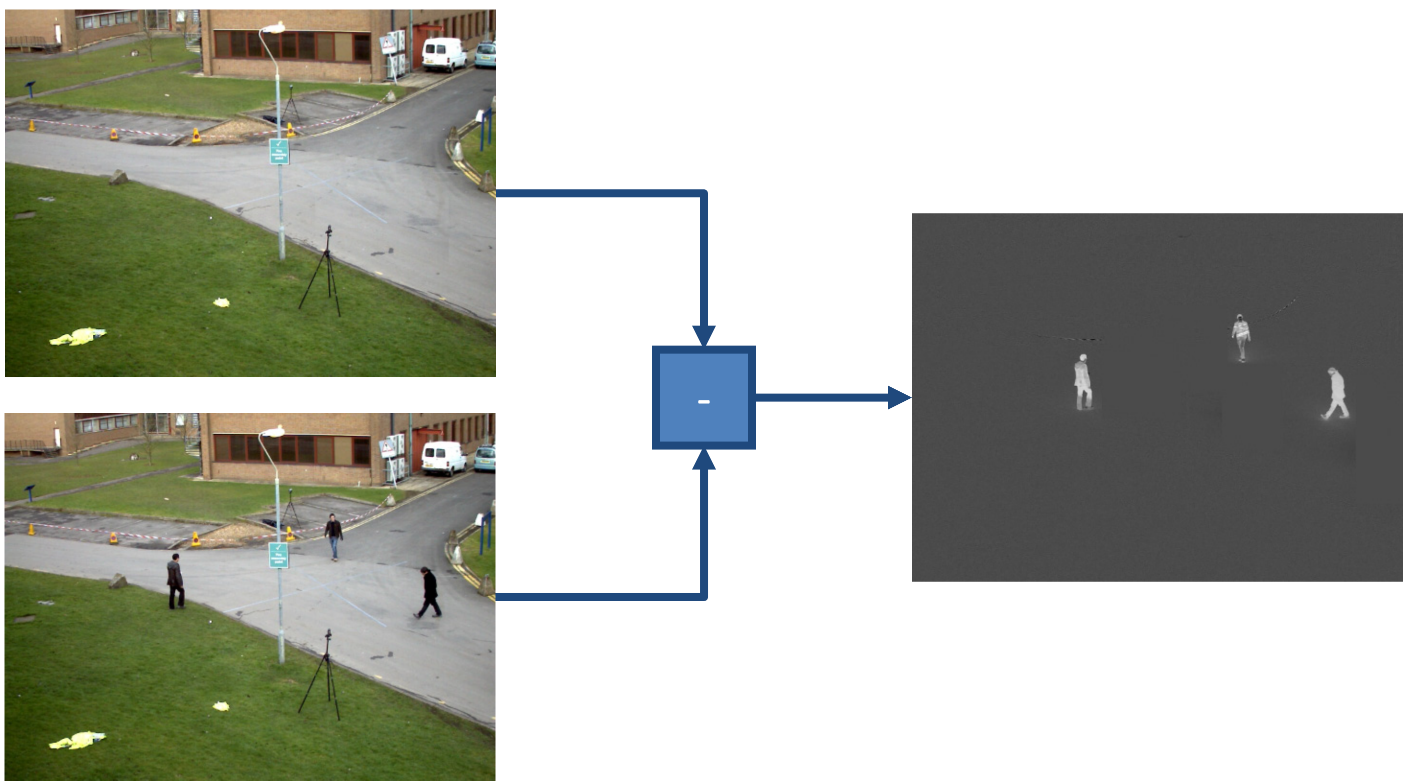
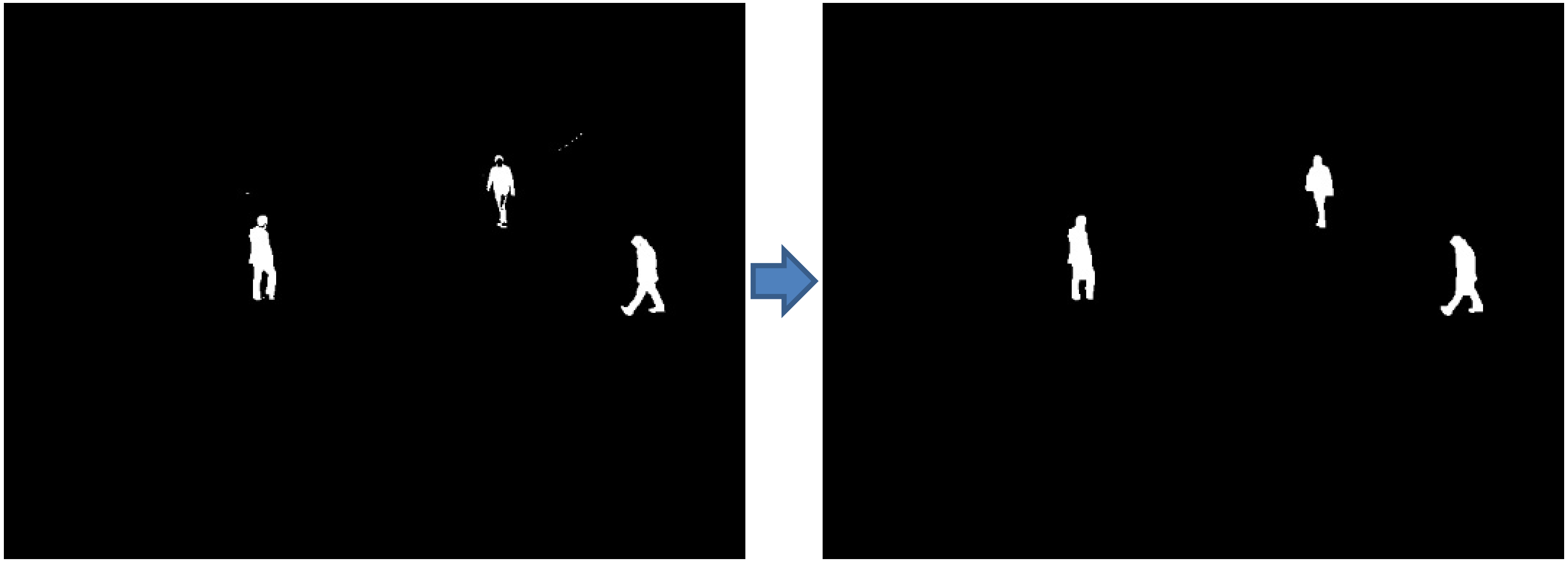


Image Subtraction

Step: Threshold and enhance the difference image to fuse neighbouring regions and remove noise



Change Detection

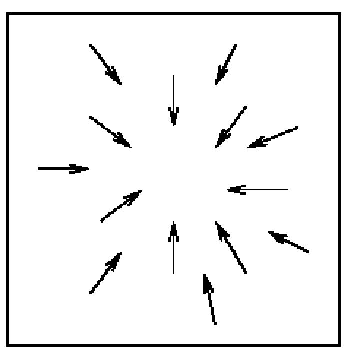
- Image subtraction algorithm
  - Input: images I_t and $I_{t-\Delta t}$ (or a model image)
  - Input: an intensity threshold τ
  - Output: a binary image I_{out}
  - Output: a set of bounding boxes B

1. For all pixels [r, c] in the input images,
   set $I_{out}[r, c] = 1$ if $(|I_t[r, c] - I_{t-\Delta t}[r, c]| > \tau)$
   set $I_{out}[r, c] = 0$ otherwise
2. Perform connected components extraction on I_{out}
3. Remove small regions in I_{out} assuming they are noise
4. Perform a closing of I_{out} using a small disk to fuse neighbouring regions
5. Compute the bounding boxes of all remaining regions of changed pixels
6. Return $I_{out}[r, c]$ and the bounding boxes B of regions of changed pixels

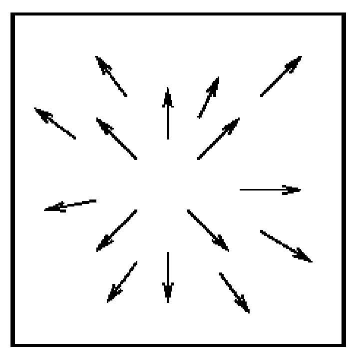
Sparse Motion Estimation

Motion Vector

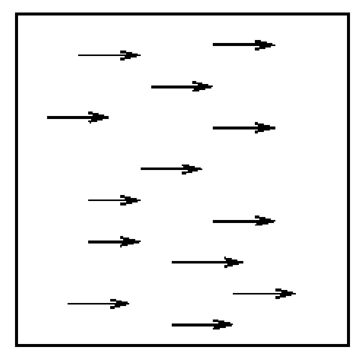
- A motion field is a 2D array of 2D vectors representing the motion of 3D scene points
- A motion vector in the image represents the displacement of the image of a moving 3D point
  - Tail at time t and head at time t+Δt
  - Instantaneous velocity estimate at time t



Zoom out



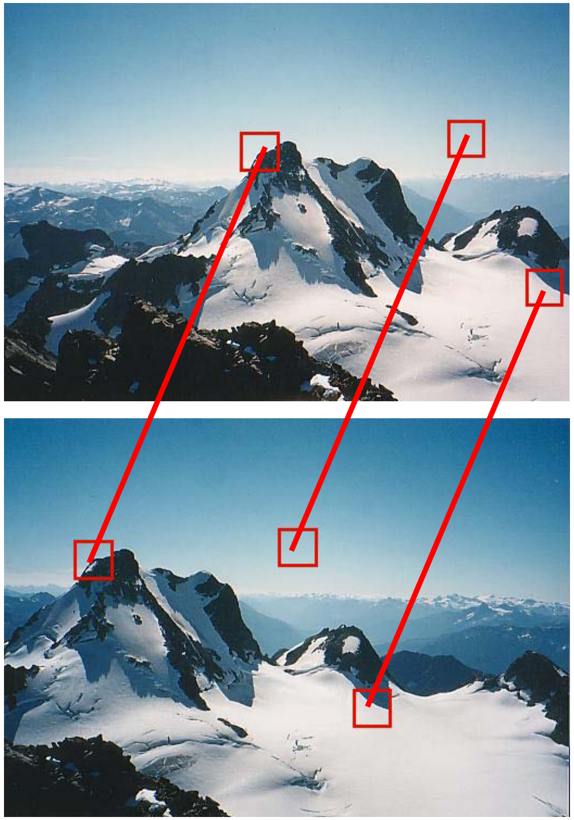
Zoom in



Pan Left

Sparse Motion Estimation

- A sparse motion field can be computed by identifying pairs of points that correspond in two images taken at times t and t+Δt
- Assumption: intensities of interesting points and their neighbours remain nearly constant over time
- Two steps:
  - Detect interesting points at t
  - Search corresponding points at t+Δt



Sparse Motion Estimation

- Detect interesting points
  - Image filters
    - Canny edge detector
    - Kirsch edge operator
    - Harris corner detector
    - SUSAN corner detector
    - Frei-Chen ripple operator
    - …
  - Interest operator
    - Computes intensity variance in the vertical, horizontal and diagonal directions
    - Interest point if the minimum of these four variances exceeds a threshold

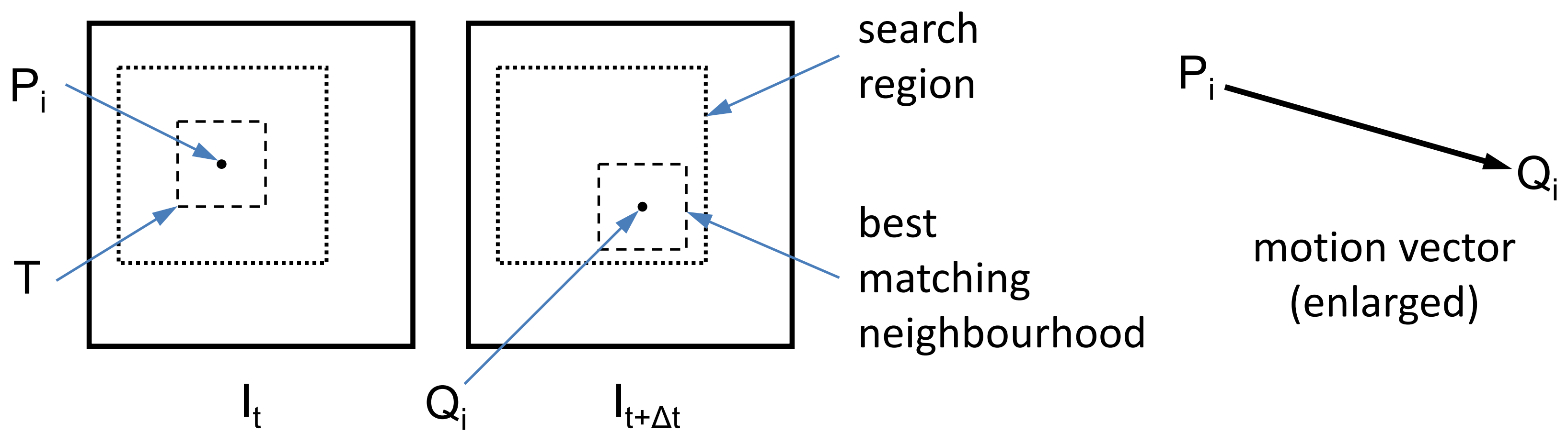
Detect Interesting Points

```
Procedure detect_interesting_points(I,V,w,t) {
    for (r = 0 to MaxRow – 1)
        for (c = 0 to MaxCol – 1)
            if (I[r,c] is a border pixel) break;
            else if (interest_operator(I,r,c,w) >= t)
                add (r,c) to set V;
}

Procedure interest_operator (I,r,c,w) {
    v1 = variance of intensity of horizontal pixels I[r,c-w]…I[r,c+w];
    v2 = variance of intensity of vertical pixels I[r-w,c]…I[r+w,c];
    v3 = variance of intensity of diagonal pixels I[r-w,c-w]…I[r+w,c+w];
    v4 = variance of intensity of diagonal pixels I[r-w,c+w]…I[r+w,c-w];
    return min(v1, v2, v3, v4);
}
```

Sparse Motion Estimation

- Search corresponding points
  - Given an interesting point P_i from I_t, take its neighbourhood in I_t and find the best matching neighbourhood in $I_{t+\Delta t}$ under the assumption that the amount of movement is limited



This approach is also known as template matching

Similarity Measures

- Cross-correlation (to be maximised)

$$\mathrm{CC}(\Delta x, \Delta y) = \sum_{(x,y)\in T} I_t\left(x, y\right) \cdot I_{t+\Delta t}\left(x + \Delta x, y + \Delta y\right)$$

- Sum of absolute differences (to be minimised)

$$\mathrm{SAD}(\Delta x, \Delta y) = \sum_{(x,y)\in T} \left| I_t\left(x, y\right) - I_{t+\Delta t}\left(x + \Delta x, y + \Delta y\right) \right|$$

- Sum of squared differences (to be minimised)

$$\mathrm{SSD}(\Delta x, \Delta y) = \sum_{(x,y)\in T} \left[I_t\left(x, y\right) - I_{t+\Delta t}\left(x + \Delta x, y + \Delta y\right) \right]^2$$

Similarity Measures

- Mutual information (to be maximised)

$$\mathrm{MI}(A,B) = \sum_{a}\sum_{b} P_{AB}(a,b)\log_2\left(\frac{P_{AB}(a,b)}{P_A(a)P_B(b)}\right)$$

Subimages to compare:

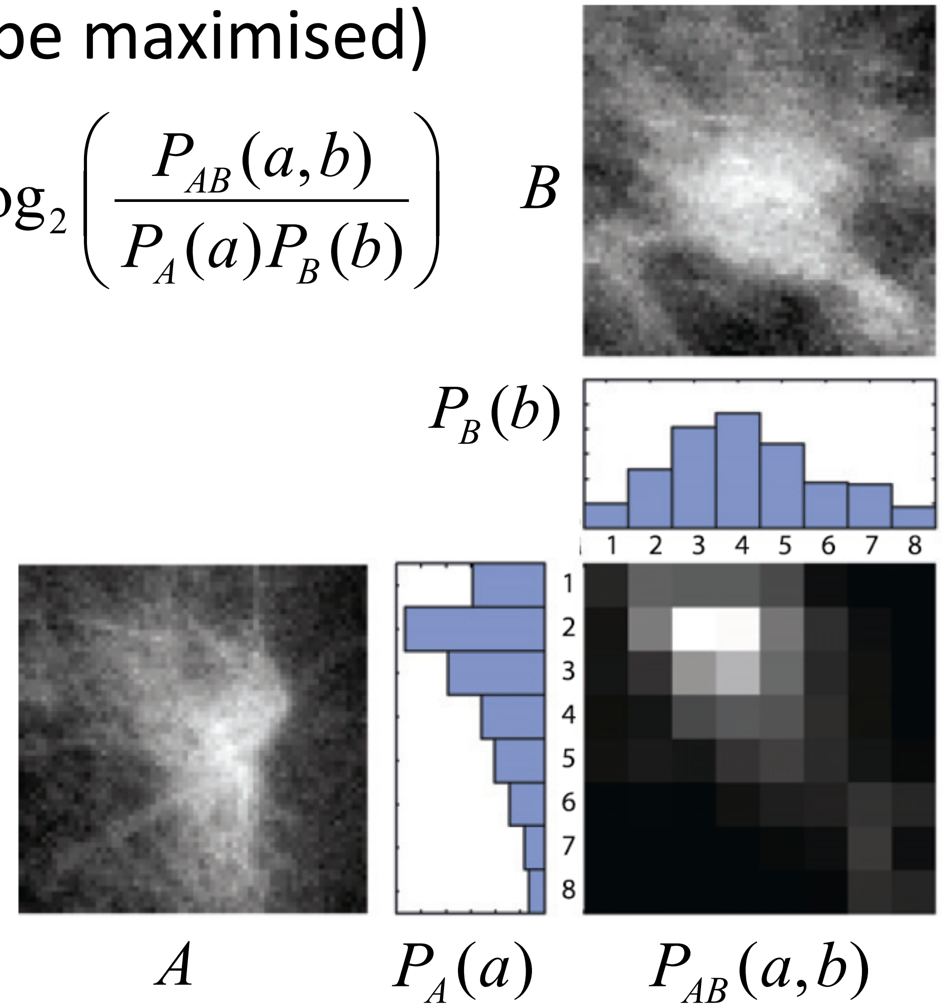
$A \subset I_t \quad B \subset I_{t+\Delta t}$

Intensity probabilities:

$P_A(a) \quad P_B(b)$

Joint intensity probability:

$P_{AB}(a,b)$



http://dx.doi.org/10.1016/j.jbi.2011.04.008

Dense Motion Estimation

Dense Motion Estimation

- Assumptions:
  - The object reflectivity and illumination do not change during the considered time interval
  - The distance of the object from the camera and the light sources do not vary significantly over this interval
  - Each small neighbourhood $N_t(x,y)$ at time t is observed in some shifted position $N_{t+\Delta t}(x+\Delta x,y+\Delta y)$ at time $t+\Delta t$
- These assumptions may not hold tight in reality, but provide useful computation and approximation

Spatiotemporal Gradient

- Taylor series expansion of a function

$$f(x+\Delta x) = f(x) + \frac{\partial f}{\partial x}\Delta x + \mathrm{h.o.t} \quad \Rightarrow$$

$$f(x+\Delta x) \approx f(x) + \frac{\partial f}{\partial x}\Delta x$$

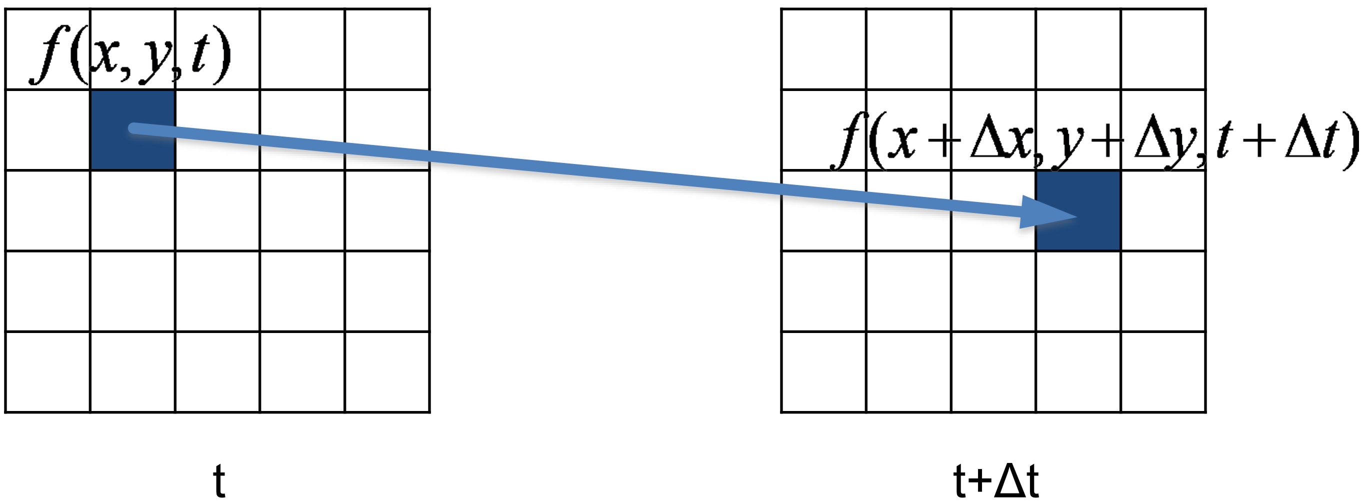
- Multivariable Taylor series approximation

$$f(x+\Delta x, y+\Delta y, t+\Delta t) \approx f(x,y,t) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y + \frac{\partial f}{\partial t}\Delta t \qquad (1)$$

Optical Flow Equation

Assuming neighbourhood $N_t(x, y)$ at time t moves over vector $V=(\Delta x, \Delta y)$ to an identical neighbourhood $N_{t+\Delta t}(x+\Delta x, y+\Delta y)$ at time $t+\Delta t$ leads to the optical flow equation:

$$f(x+\Delta x, y+\Delta y, t+\Delta t) = f(x, y, t) \qquad (2)$$



Optical Flow Computation

Combining (1) and (2) yields the following constraint:

$$\frac{\partial f}{\partial x}\Delta x+\frac{\partial f}{\partial y}\Delta y+\frac{\partial f}{\partial t}\Delta t=0\Rightarrow$$

$$\frac{\partial f}{\partial x}\frac{\Delta x}{\Delta t}+\frac{\partial f}{\partial y}\frac{\Delta y}{\Delta t}+\frac{\partial f}{\partial t}\frac{\Delta t}{\Delta t}=0\Rightarrow$$

$$\frac{\partial f}{\partial x}v_x+\frac{\partial f}{\partial y}v_y+\frac{\partial f}{\partial t}=0\Rightarrow$$

$$\nabla f\cdot v=-f_t$$

where $v=(v_x, v_y)$ is the velocity or *optical flow* of $f(x,y,t)$
and $\nabla f=(f_x, f_y)=(\partial f/\partial x,\ \partial f/\partial y)$ is the gradient

Optical Flow Computation

- The optical flow equation provides a constraint that can be applied at every pixel position
- However, the equation does not have unique solution and thus further constraints are required

For example, by using the optical flow equation for a group of adjacent pixels and assuming that all of them have the same velocity, the optical flow computation task amounts to solving a linear system of equations using the least-squares method

Many other solutions have been proposed (see references)

Optical Flow Computation

- Example: Lucas-Kanade approach to optical flow

  Assume the optical flow equation holds for all pixels p_i in a certain neighbourhood and use the following notation:

$$v = (v_x , v_y) \qquad f_x = \frac{\partial f}{\partial x} \qquad f_y = \frac{\partial f}{\partial y} \qquad f_t = \frac{\partial f}{\partial t}$$

  Then we have the following set of equations:

$$\begin{aligned}
f_x(p_1)v_x + f_y(p_1)v_y &= -f_t(p_1) \\
f_x(p_2)v_x + f_y(p_2)v_y &= -f_t(p_2) \\
\vdots \qquad\qquad \vdots \qquad & \qquad \vdots \\
f_x(p_N)v_x + f_y(p_N)v_y &= -f_t(p_N)
\end{aligned}$$

Optical Flow Computation

- Example: Lucas-Kanade approach to optical flow

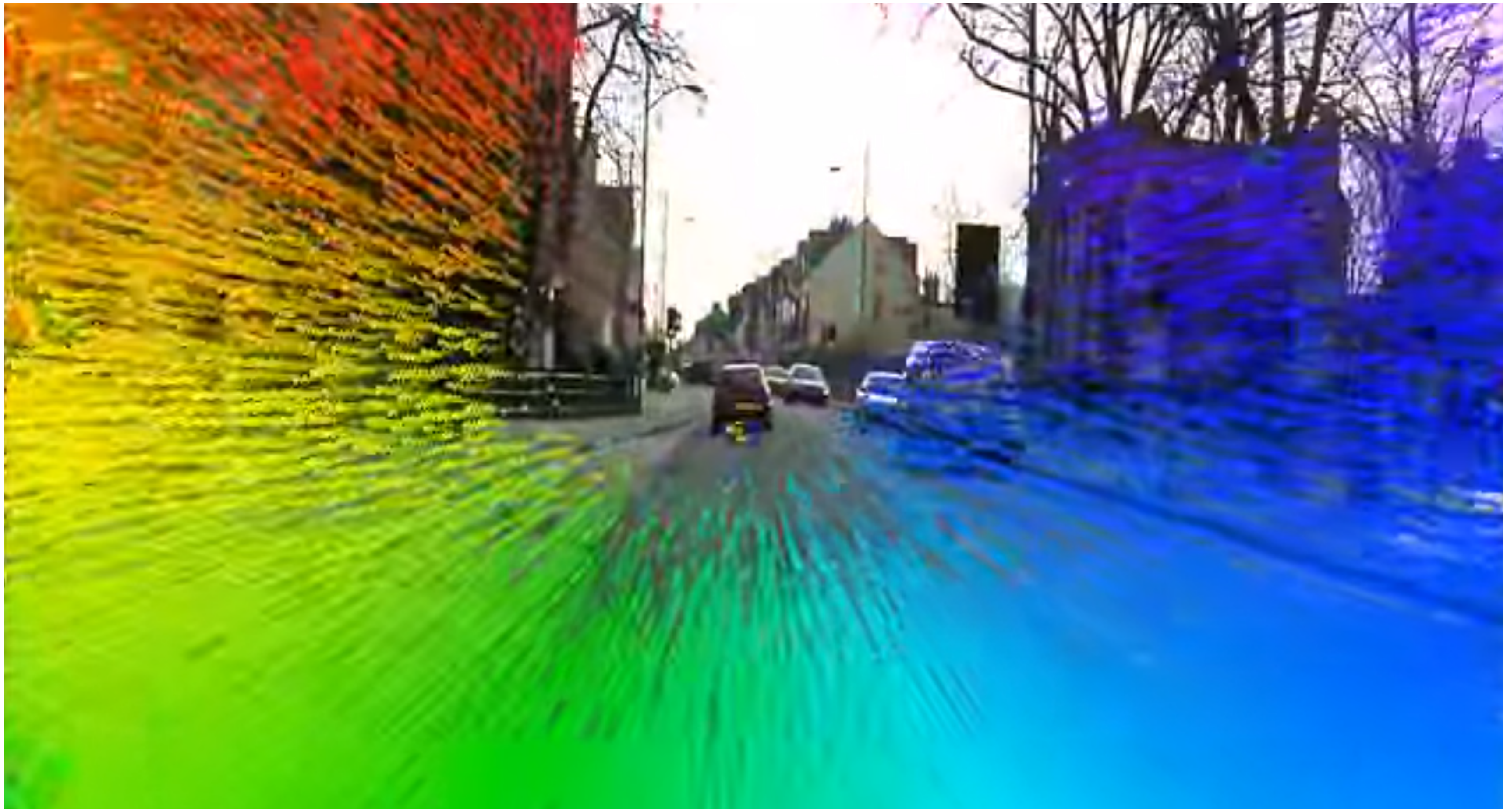
The set of equations can be rewritten as $Av = b$ where

$$A = \begin{bmatrix} f_x(p_1) & f_y(p_1) \\ f_x(p_2) & f_y(p_2) \\ \vdots & \vdots \\ f_x(p_N) & f_y(p_N) \end{bmatrix} \qquad v = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \qquad b = \begin{bmatrix} -f_t(p_1) \\ -f_t(p_2) \\ \vdots \\ -f_t(p_N) \end{bmatrix}$$

This can be solved using the least-squares approach:

$$A^T A v = A^T b \quad \Rightarrow \quad v = \left(A^T A \right)^{-1} A^T b$$

Optical Flow Example



https://www.youtube.com/watch?v=GIUDAZLfYhY

References and Acknowledgements

- Chapter 8 of Szeliski 2010
- Chapter 9 of Shapiro and Stockman 2001
- Images drawn from the above references