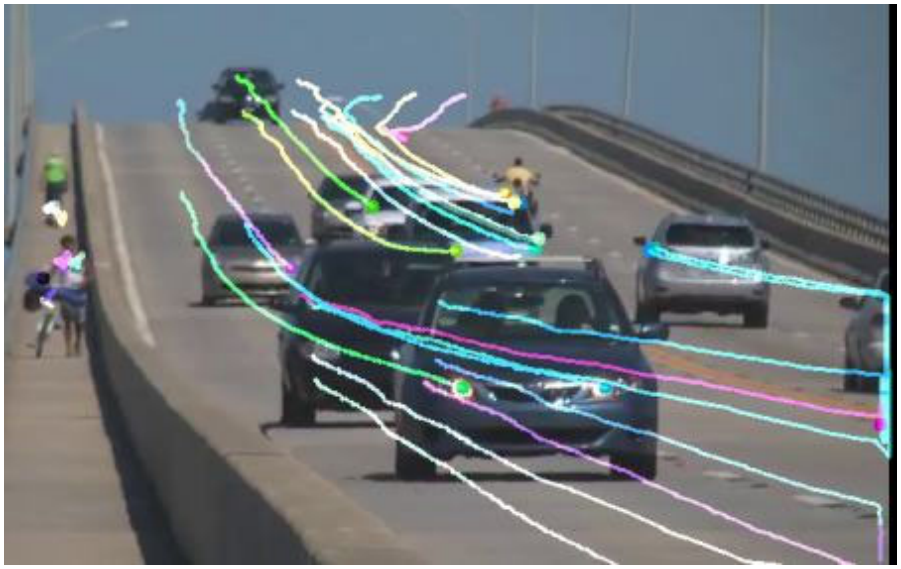
Motion

Computer Vision and Pattern Recognition

CS 4243

S1-Y2023/24

NUS National University of Singapore | School of Computing



Lesson 5

Motion Detection and Optical Flow

Contents

- Constraints
- Differential algorithms
- Optical flow
- Object tracking
- Background modeling
- Kalman filters
- Particle filters

Introduction

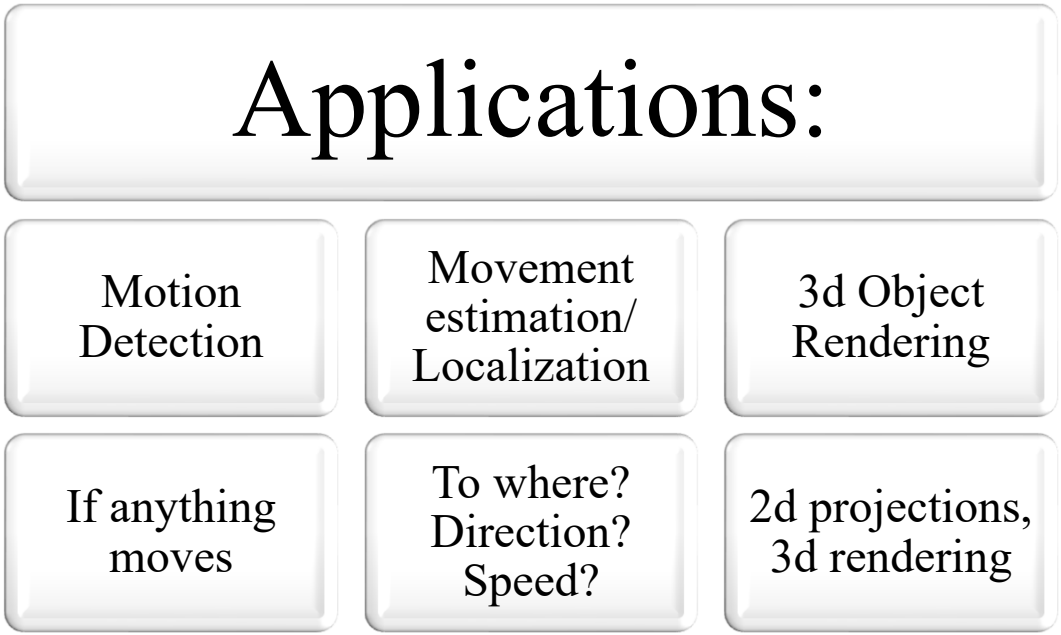
- Increasing processing power and, especially, the wide deployment of surveillance technology have made the automated study of motion desirable and possible.
- Detection and tracking of human faces, pedestrian or vehicular motion are now common applications; additionally, object-based video compression, driver assistance, autonomous vehicles, robot navigation, user interfaces, smart room tracking, etc. can be seen.

Introduction

- Increasing processing power and, especially, the wide deployment of surveillance technology have made the automated study of motion desirable and possible.
- Detection and tracking of human faces, pedestrian or vehicular motion are now common applications; additionally, object-based video compression, driver assistance, autonomous vehicles, robot navigation, user interfaces, smart room tracking, etc. can be seen.

Introduction

- **There are three main groups of motion-related problems from the practical point of view:**
  1. **Motion detection**
  2. **Moving object detection and localization**
  3. **Derivation of 3D object properties**



Basic Assumptions

Maximum velocity	Small acceleration	Common motion	Mutual correspondence
Speed of the moving objects are not high compared to our FPS rate.	Acceleration is almost 0	Objects are rigid and a single motion vector is applicable on all parts of the object	Rigid objects exhibit stable pattern points.

Basic Assumptions

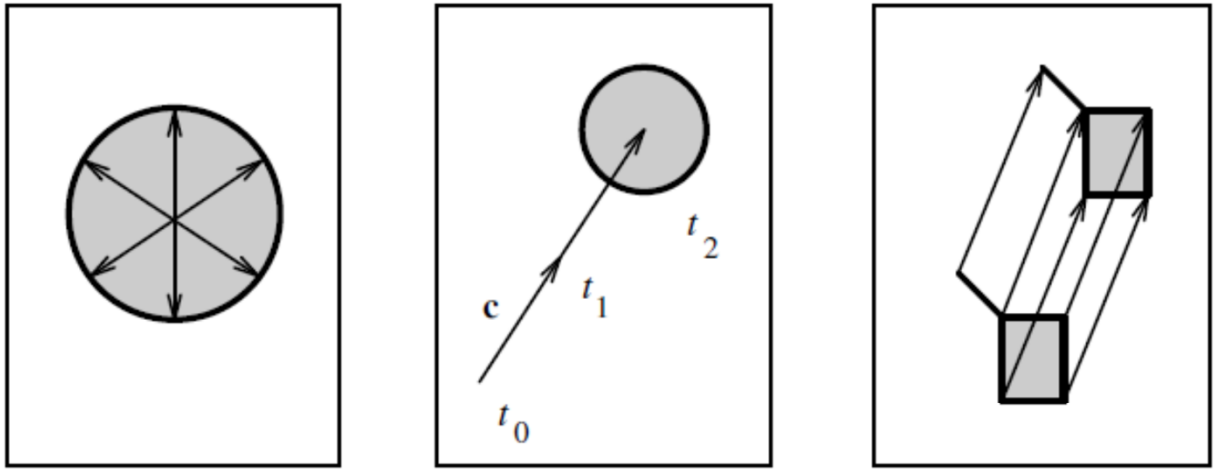


Figure 16.1: Object motion assumptions. (a) Maximum velocity (shaded circle represents area of possible object location). (b) Small acceleration (shaded circle represents area of possible object location at time t_2). (c) Common motion and mutual correspondence (rigid objects).

Generally,

To generalize, image motion analysis and especially object tracking combine two separate but interrelated components:

Localization and representation of the object of interest (target)

Trajectory filtering and data association

Differential Motion Analysis Methods

- Simple subtraction of images acquired at different instants in time makes motion detection possible, assuming a stationary camera position and constant illumination.
- $$d(i,j) = \begin{cases} 0 & if\ |f_1(i,j) - f_2(i,j)| < \epsilon \\ 1 & otherwise \end{cases}$$
- Sometimes: *f(I,j,t)*
- d is the difference image
- f_1 *and* f_2 *are* two consecutive images separated by a time interval, i.e., 2 consecutive frames.
- Q: when d(i,j) is 1?

Differential Motion Analysis Methods

- Simple subtraction of images acquired at different instants in time makes motion detection possible, assuming a stationary car
- $d($
- So
- d i
- f_1 c
  int
- Q: when d(i,j) is 1?

- $f1(i, j)$ is a pixel on a moving object, $f2(i, j)$ is a pixel on the static background, (or vice versa).
- $f1(i, j)$ is a pixel on a moving object, $f2(i, j)$ is a pixel on another moving object.
- $f1(i, j)$ is a pixel on a moving object, $f2(i, j)$ is a pixel on a different part of the same moving object.
- Noise, inaccuracies of stationary camera positioning, etc.

Differential Motion Analysis Methods

- **Cumulative Difference Images:**
- $\boldsymbol{d_{cum}(i,j) = \sum_{k=1}^{n} a_k \, |f_1(i,j) - f_k(i,j)|}$
- Sometimes *f(i,j,k)*
- *ak* gives the *significance* of images in the sequence of *n* images; more recent images may be given greater weights to reflect the importance of current motion and to specify current object location.

$$a_k = \frac{k}{n}$$

Differential Motion Analysis Methods

jupyter motion_det1 Last Checkpoint: 18/09/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

motion detection example 1

amir 2023

In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

In [2]:
```
def ready_2_show(a, level=255):
    a = ( a - np.min(a) ) / (np.max(a) - np.min(a))
    a = a * level
    return np.uint8(a)
```

Differential motion detection

In [3]:
```
cap = cv2.VideoCapture("..\\10231.mp4")
print( type(cap) )
ret, frame1 = cap.read()
N=13
for i in range(N):
    ret , frtmp = cap.read()
ret, frame2 = cap.read()
```

```
<class 'cv2.VideoCapture'>
```

Issues

- **Motion field**
- **Aperture problem**

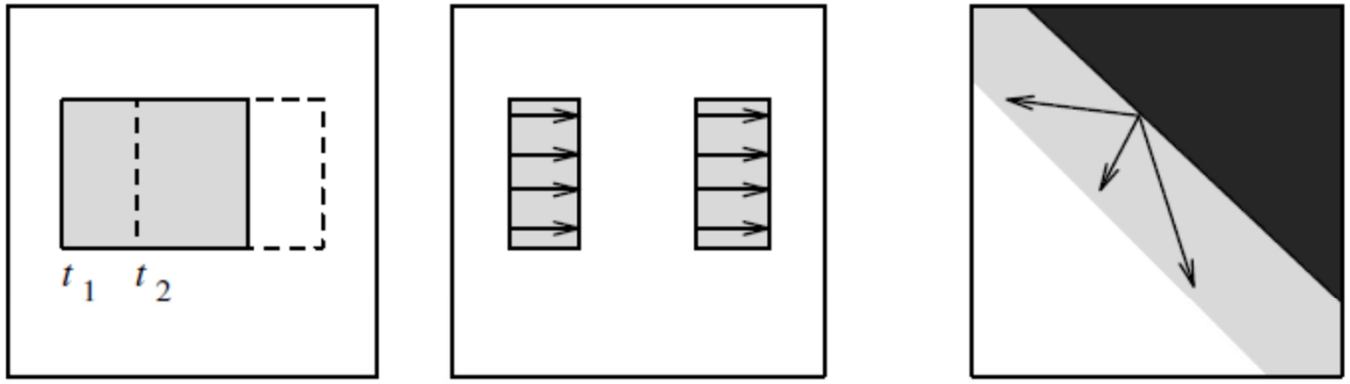


Figure 16.4: Problems of motion field construction. (a) Object position at times t_1 and t_2. (b) Motion field. (c) Aperture problem—ambiguous motion. © *Cengage Learning 2015.*

- **Solution**: Detecting moving edges helps overcome several limitations of differential motion analysis methods.

Hypothesis

- Moving edges can be determined by logical AND operations of the spatial and temporal image edges.
- The spatial edges can be identified by virtually any edge detector. The temporal gradient can be approximated using the difference image, and the logical AND can be implemented through multiplication.
- Then, the moving edge image $d_{\text{med}}(i, j)$ can be determined as:

$$d_{med}(i,j) = S(i,j).D(i,j)$$

S(i,j)= edge magnitude of one of the frames

D(i,j)= absolute difference image

Hypothesis

Figure 16.5: Moving-edge image determined from the first and second frames of the image sequence analyzed in Figure 16.2 (inverted for improved visualization). © *Cengage Learning 2015.*

Edge Detectors

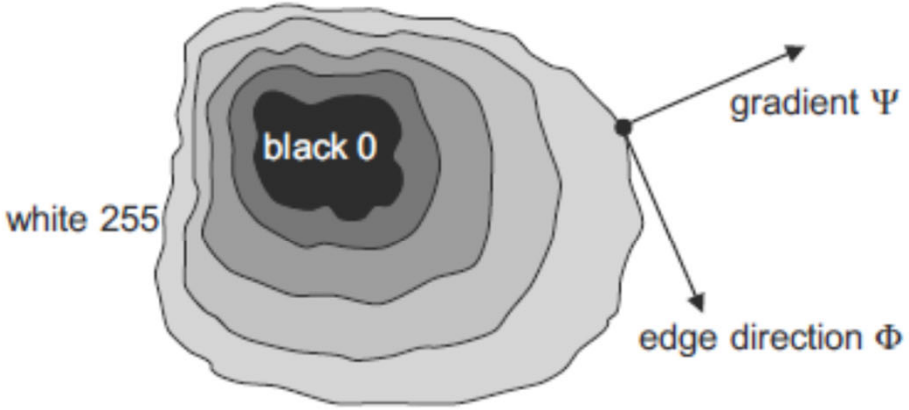


Figure 5.16: Gradient direction and edge direction. © *Cengage Learning 2015.*

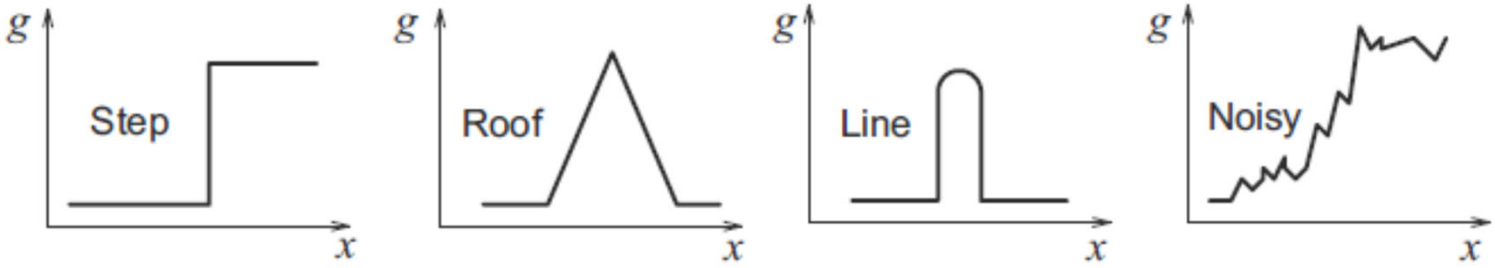


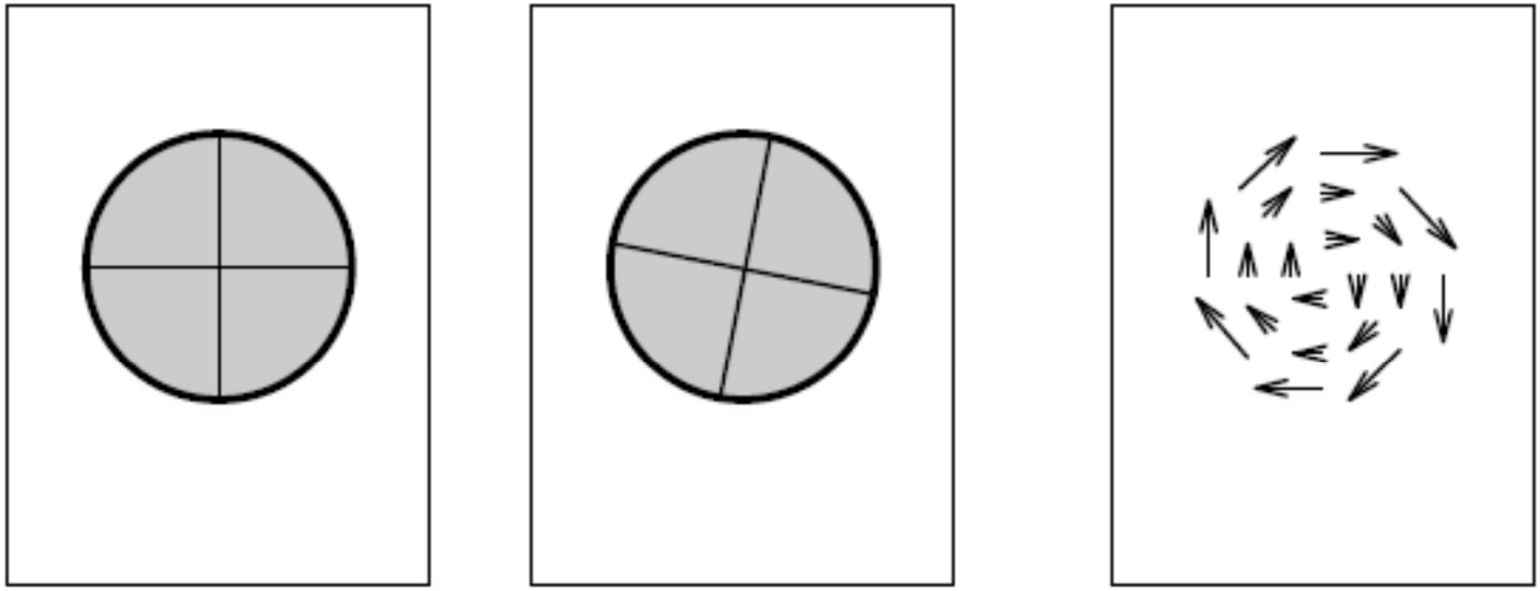
Figure 5.17: Typical edge profiles. © *Cengage Learning 2015.*

Edge Detectors: e.g. Roberts, Laplace, Prewitt, Sobel, (See S_IPAMV4 5.3.2)

Optical Flow

- Optical flow reflects image changes due to motion during a time interval dt, and the optical flow field is the velocity field that represents the three-dimensional motion of object points across a two-dimensional
- It is an abstraction
- It should represent only those motion-related intensity changes in the image that are required in further processing
- Its computation is a necessary precondition of subsequent higher-level processing that can solve motion-related problems

Optical Flow



Optical flow. (a) Time t_1. (b) Time t_2. (c) Optical flow.

Optical Flow Algorithm

- **Assumptions:**

1- The observed brightness of any object point is constant over time.

2- Nearby points in the image plane move in a similar manner (the *velocity smoothness* constraint).

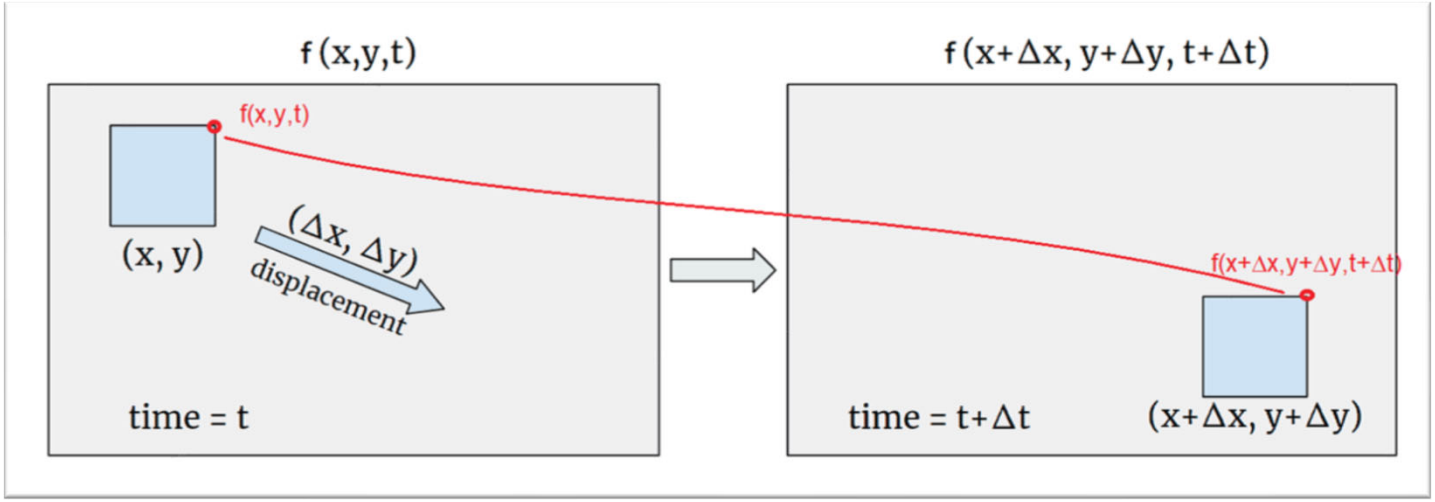
- **Algorithm:**
  1. we have a continuous image; f(x, y, t) refers to the gray-level of (x, y) at time t. Representing a dynamic image as a function of position and time permits it to be expressed as a Taylor series:

$$f(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t) = f(x, y, t) + f_x \mathrm{d}x + f_y \mathrm{d}y + f_t \mathrm{d}t + O(\partial^2)$$

Optical Flow Algorithm

2. fx, fy, ft denote the partial derivatives of f. We can assume that the immediate neighborhood of (x, y) is translated some small distance (dx, dy) during the interval dt; that is, we can find dx, dy, dt such that:

$$f(x + \mathrm{d}x, y + \mathrm{d}y, t + \mathrm{d}t) = f(x, y, t) \; .$$



Optical Flow Algorithm

3. If we ignore 2nd and higher order terms in the Taylor series, then 1 and 2 yield:

$$f_x dx + f_y dy + f_t dt = 0$$

Or

$$-f_t = f_x \frac{dx}{dt} + f_y \frac{dy}{dt}$$

4. Goal is to compute the velocity:

$$\mathbf{c} = \left(\frac{\mathrm{d}x}{\mathrm{d}t}, \frac{\mathrm{d}y}{\mathrm{d}t} \right) = (u, v)$$

Optical Flow Algorithm

5. If we can approximate fx, fy, ft, then ***c*** can be determined:

$$-f_t = f_x u + f_y v = \Delta f \,.\, c$$

$$c = (f_x u + f_y v)/\Delta f$$

Δf is a two-dimensional image gradient.

6. After many mathematics, e.g. using smoothness constraint, the problem would be converted to minimization of the squared error term below:

$$E^2(x,y) = \left(f_x u + f_y v + f_t\right)^2 + \lambda\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right)$$

Optical Flow Algorithm

5. Finally, it reduces to solving these differential equations:

$$\left(\lambda^2 + f_x^2\right) u + f_x f_y v = \lambda^2 \overline{u} - f_x f_t \ ,$$

$$f_x f_y u + \left(\lambda^2 + f_y^2\right) v = \lambda^2 \overline{v} - f_y f_t \ ,$$

λ is the Lagrange coefficient, and e.g. u_x^2 denotes partial derivatives squared as error terms.

However, the algorithm is not that complicated.

Optical Flow Algorithm

Algorithm 16.1: Relaxation computation of optical flow from dynamic image pairs

1. Initialize velocity vectors $\mathbf{c}(i,j) = 0$ for all (i,j).
2. Let k denote the number of iterations. Compute values u^k, v^k for all pixels (i,j)

$$u^k(i,j) = \overline{u}^{k-1}(i,j) - f_x(i,j)\frac{P(i,j)}{D(i,j)}\,,$$
$$v^k(i,j) = \overline{v}^{k-1}(i,j) - f_y(i,j)\frac{P(i,j)}{D(i,j)}\,. \qquad (16.14)$$

Partial derivatives f_x, f_y, f_t can be estimated from pairs of consecutive images.

3. Stop if

$$\sum_i \sum_j E^2(i,j) < \varepsilon\,,$$

where ε is the maximum permitted error; return to step 2 otherwise.

If more than two images are to be processed, efficiency may be increased by using the results of one iteration to initialize the current image pair in the sequence.

Optical Flow Applications

Motion, as it appears in dynamic images, is usually some combination of four basic elements:

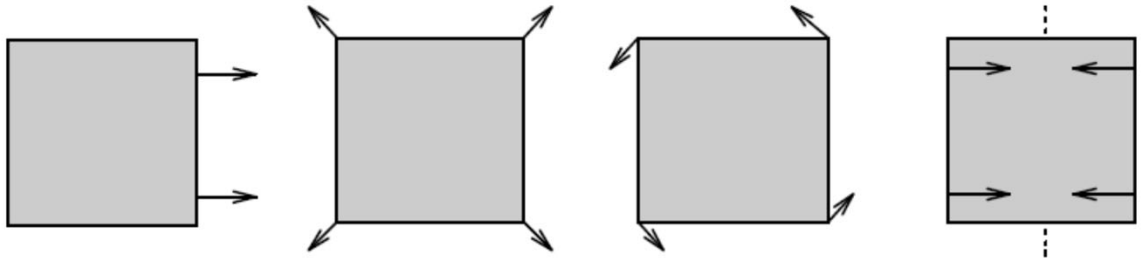
- **Translation at a constant distance from the observer.**
- **Translation in depth relative to the observer.**
- **Rotation at a constant distance about the view axis.**
- **Rotation of a planar object perpendicular to the view axis.**

Optical Flow Applications

Optical-flow-based motion analysis can recognize these basic elements by applying a few relatively simple operators to the flow.

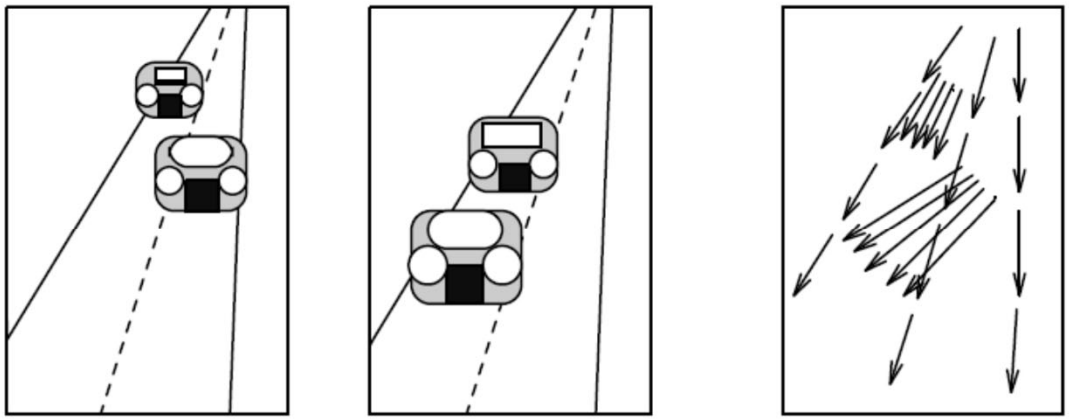
- **Translation at a constant distance is represented as a set of parallel motion vectors.**
- **Translation in-depth forms a set of vectors having a common focus of expansion.**
- **Rotation at constant distance results in a set of concentric motion vectors.**
- **Rotation perpendicular to the view axis forms one or more sets of vectors starting from straight line segments.**

Optical Flow Applications



(a) (b) (c) (d)

Figure 16.9: Motion form recognition. (a) Translation at constant distance. (b) Translation in depth. (c) Rotation at constant distance. (d) Planar object rotation perpendicular to the view axis. © *Cengage Learning 2015.*



(a) (b) (c)

Figure 16.10: Focus of expansion. (a) Time t_1. (b) Time t_2. (c) Optical flow. © *Cengage Learning 2015.*

Optical Flow Algorithm

jupyter optical_flow4 Last Checkpoint: 19/09/2023 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import cv2
import numpy as np

# The video feed is read in as
# a VideoCapture object
cap = cv2.VideoCapture("..\\10236.mp4")
```

In [2]:
```
ret, old_frame = cap.read()
old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)
p0 = cv2.goodFeaturesToTrack(old_gray, maxCorners=100, qualityLevel=0.3, minDistance=7)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Calculate optical flow
    lk_params = dict(winSize=(15, 15), maxLevel=2, criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
    p1, st, err = cv2.calcOpticalFlowPyrLK(
        old_gray, frame_gray, p0, None, **lk_params
    )

    # Select good points
    good_new = p1[st == 1]
    good_old = p0[st == 1]

    # Draw optical flow vectors
    for i, (new, old) in enumerate(zip(good_new, good_old)):
        a, b = new.ravel()
        c, d = old.ravel()
        cv2.line(frame, (int(a), int(b)), (int(c), int(d)), (0, 255, 0), 2)
        cv2.circle(frame, (int(a), int(b)), 5, (0, 0, 255), -1)
```

Object Tracking

- **An algorithm detects objects and then tracks their movements in space or across different camera angles. Object tracking can identify and follow single or multiple objects in videos.**
- **There are four stages of object tracking:**
  1. Target initialization.
  2. Appearance modeling.
  3. Motion estimation.
  4. Target positioning.

Interest Points / Anchor Points

- **AKA Feature Points**
- **They help us with object tracking**
- **There is a host of interest point detectors that have developed over the years (see 5.3.10):**
  - Moravec
  - Kitchen-Rosenfeld
  - Zuniga–Haralick
  - Harris Corner Detector

Lucas-Kanade Point Tracking Algorithm

Algorithm 16.3: General Lucas–Kanade tracking

1. Initialize a warp $\mathbf{W}(\mathbf{x};\mathbf{p})$, determined by some parameters $\mathbf{p}$.
2. Warp I with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to $\hat{I}$, and determine the error $T - \hat{I}$.
3. Warp the gradient of I, ∇I with $\mathbf{W}(\mathbf{x};\mathbf{p})$, and evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$. Compute the 'steepest descent' image $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, and the *Hessian* matrix

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

4. Compute

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

5. Set

$$\mathbf{p} = \mathbf{p} + \Delta \mathbf{p}$$

and go to 2 until $||\Delta \mathbf{p}|| < \epsilon$.

Lucas-Kanade Point Tracking Algorithm

jupyter lk1 Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python

Run Markdown

Lucas Kanade Example

In this code:

1. We start by capturing video frames from a video file (you can also use a camera).
2. We track feature points using the Lucas-Kanade algorithm in the lucas_kanade_point_tracking function.
3. We draw the tracked points and the motion vectors on each frame.
4. The tracking continues until you press the 'Esc' key to exit the application.

- Make sure to replace 'video.mp4' with the path to your video file or configure the code to use your camera as the video source.
- You may also need to adjust the parameters for good feature point detection (cv2.goodFeaturesToTrack) to suit your specific tracking needs.

In [1]:
```
import cv2
import numpy as np
```

In [2]:
```
def lucas_kanade_point_tracking(prev_frame, curr_frame, prev_points):
    # Parameters for Lucas-Kanade optical flow
    lk_params = dict(winSize=(15, 15),
                     maxLevel=2,
                     criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))

    # Calculate optical flow using Lucas-Kanade
    next_points, status, _ = cv2.calcOpticalFlowPyrLK(prev_frame, curr_frame, prev_points, None, **lk_params)

    # Filter out the points with status=1 (successfully tracked)
    prev_points = prev_points[status == 1]
    next_points = next_points[status == 1]

    return prev_points, next_points
```

Background Modeling

1. **Q. What is moving in the scene?**
2. **A. Comparing what we see with an empty, background scene, and performing some sort of comparison or subtraction.**
3. **In a video, the first frame is the background**
4. **Then, in the next frames, we derive blobs that will be the objects moving in the scene.**
5. **Disadvantages:**
    1. Wind shaking trees,
    2. The tiniest camera movement
    3. Any change in the environment

Background Modeling

- **We might concede that 'background' is not constant and try to maintain a background frame dynamically.**
- **Success has been had with this approach by taking the mean or (more usually) median intensity at each pixel of the last K frames.**
- **This will have the effect, over a period K, of allowing a per-pixel update of lighting.**

Background Maintenance by Median Filtering Algorithm

1. Initialize: Acquire K frames. At each pixel, determine the median intensity of these K. If the image sequence is in color, this is done for each of the R,G,B streams. The median represents the current background value.
2. Acquire frame $K+1$. Compute the difference between this frame and the current background at each pixel (a scalar for a gray image; a vector for RGB).
3. Threshold this difference to remove/reduce noise. Simple thresholds may be too crude at this point and there is scope for using, e.g., hysteresis (Algorithm 6.5).
4. Use some combination of blurring and morphological operations (Sections 13.3.1 and 13.3.2) to remove very small regions in the difference image, and to fill in 'holes' etc. in larger ones. Surviving regions represent the moving objects.
5. Update the median measurement, incorporating frame $K+1$ and discarding the current earliest measurement.
6. Return to (2) for the next frame.

Background Maintenance by Median Filtering Algorithm

Figure 16.16: Left—a pedestrian scene; right—movement extracted from the scene by maintaining a median-filtered background. Post-processing has been applied to remove 'small' blobs and to tidy up the figure boundary. *Courtesy of A. M. Baumberg, University of Leeds.*

Next: Mixture of Gaussian

- The idea is to model each pixel independently, in each case as a mixture of Gaussians
- Depending on local behavior, some of these Gaussians will represent the background and some foreground: the algorithm provides a means of deciding which.
- Advantages:
  - If each pixel were the result of a particular surface under invariant lighting, a single Gaussian would be sufficient to model its appearance.
  - If the lighting were varying slowly over time, then a single adapting Gaussian would be sufficient.

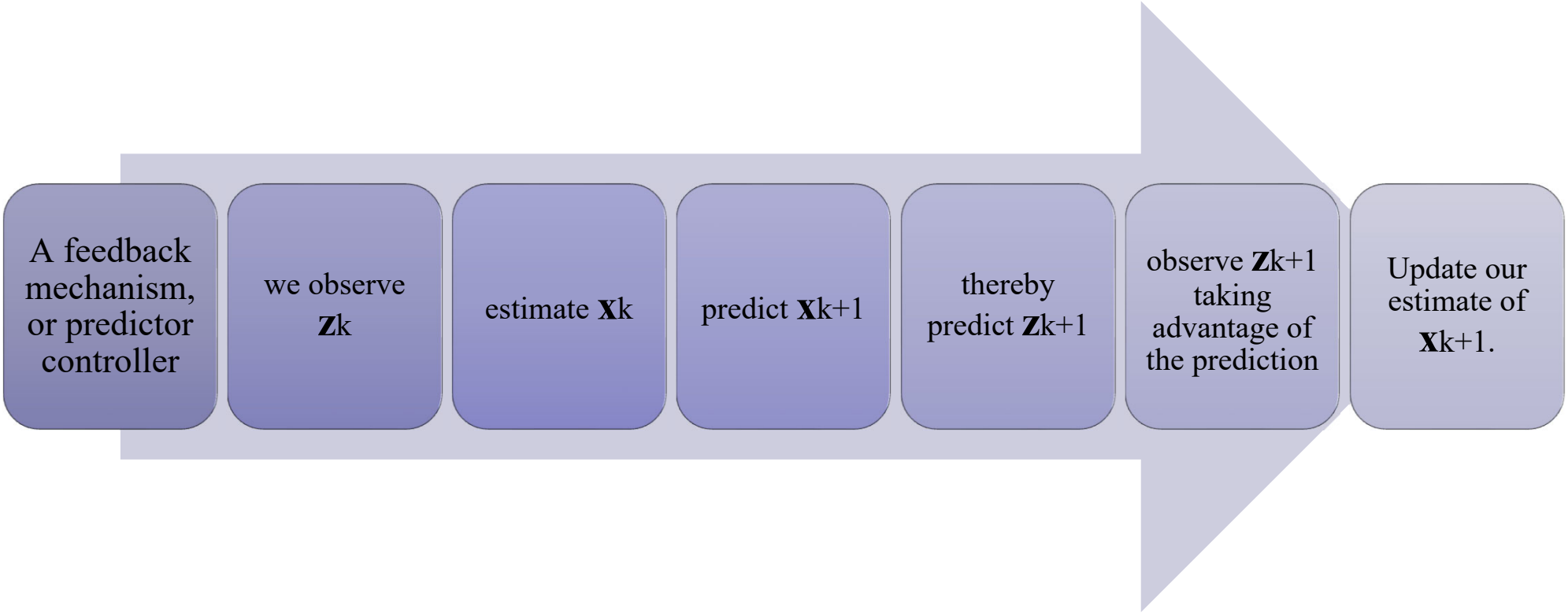
Motion Models to Aid Tracking

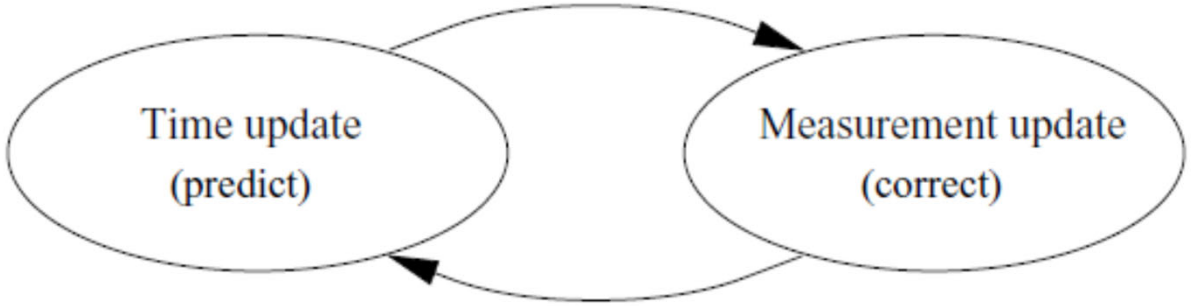
- Tracking: Where is it after Δt? Position in a 2d/3d space
- Control theory: the estimation of the state of a time-varying system via a sequence of noisy measurements. State in the state space
- Gauss 1809, to determine the orbits
- of celestial bodies
- Formally, we may have a model of an object in a scene that we see through noise;
- Thus we model x and observe z, where x and z are feature vectors

Motion Models to Aid Tracking

- In the circumstance of observing through an image sequence, in particular motion, we may be able to use the models and observations in two ways:
  - Multiple observations z_1, z_2, . . . should permit an improved estimate of the underlying model x. z_k will give an estimate of x_k; provided we have a clear understanding of how x_k changes
  - The estimate of x at time k may also provide a prediction for the observation x_{k+1}, and thereby for z_{k+1}.

Motion Models to Aid Tracking

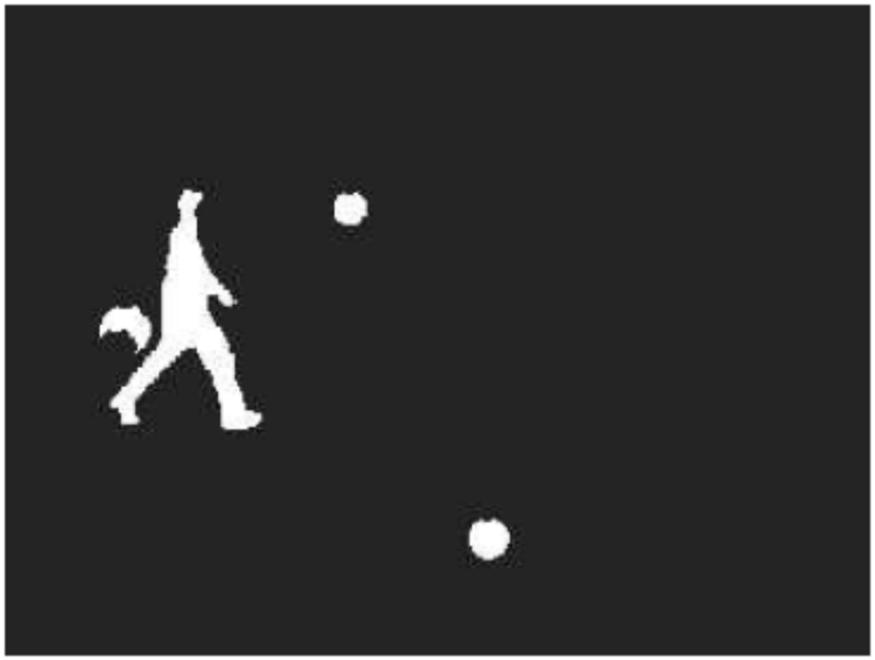


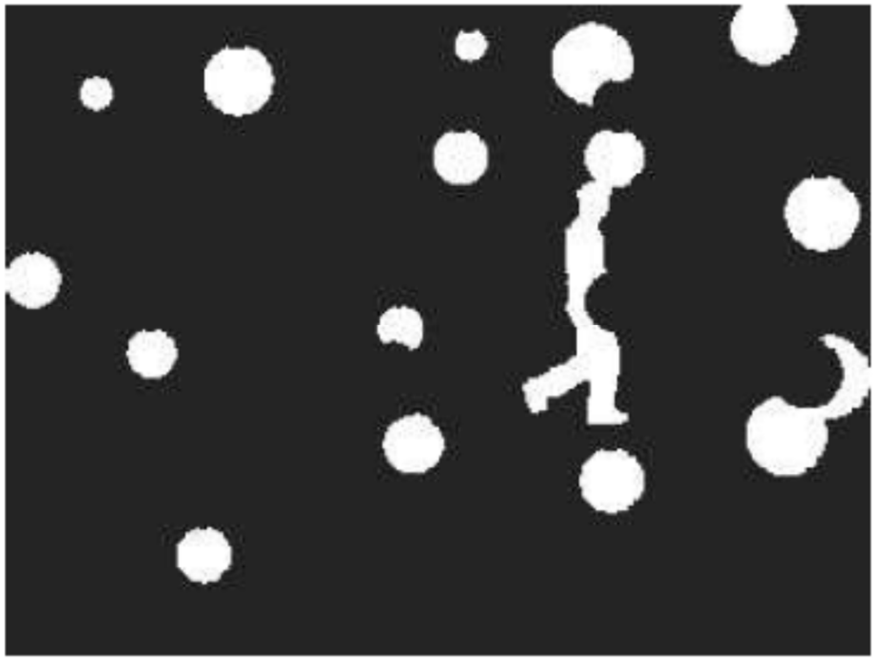


Kalman Filters

- The Kalman filter is key to providing real-time performance in this application.
- Estimates of where the silhouette ought to be permit very localized search for edges in new frames, obviating the need for operations over the whole image.
- This operation is performed by determining the edge normal at the predicted position, and searching along it for the position of maximum contrast with the reference background.
- If no likely point is found (the contrast is low), ‘no observation’ is recorded, and the associated predictions are not updated.
- This is particularly powerful since it permits the figure to suffer partial occlusion and still be tracked

Kalman Filters





Tracking through occlusion using Kalman filter

Particle Filter

- **Kalman filters disadvantages:**
  - Tracking through substantial noise and clutter in real-time
  - the assumption of local unimodal, Gaussian distributions is often invalid.
- **→ We need a more general approach using particle filters in which systems are represented by sets of probabilistically derived samples that provide an empirical description of what is and is not 'likely'.**
- **Interestingly, this more general (and powerful) approach can be made to run satisfactorily in real-time, and has a more accessible analysis than the Kalman filter.**

Particle Filter

Algorithm 16.8: Condensation (particle filtering)

1. Assume a weighted sample set at time $t-1$ is known

$$S_{t-1} = \left\{ (\mathbf{s}_{(t-1)i}, \pi_{(t-1)i}) \right\}, \quad i = 1, 2, \ldots, N .$$

Set

$$\begin{aligned} c_0 &= 0 , \\ c_i &= c_{i-1} + \pi_{(t-1)i} , \quad i = 1, 2, \ldots, N , \end{aligned} \tag{16.96}$$

(the cumulative probabilities).

2. To determine the n^{th} sample of S_t, select a random number in the range [0,1], and determine $j = \text{argmin}_i(c_i > r)$; we shall propagate sample j. This is called *importance sampling*, a sampling technique that weights towards the more probable.

3. *Prediction* (Figure 16.26): Use knowledge of the Markovian behavior of $\mathbf{x}_t$ to derive $\mathbf{s}_{tn}$. How precisely this is done depends upon the Markov relationship: in the Kalman case, we would have

$$\mathbf{s}_{tn} = A_{t-1}\mathbf{s}_{(t-1)j} + \mathbf{w}_{t-1}$$

for matrices A_{t-1} and noise $\mathbf{w}_{t-1}$, but this relationship is not constrained. Importantly, note that $\mathbf{s}_{(t-1)j}$ may well be selected more than once as we iterate from (2), but this propagation may be expected to generate different $\mathbf{s}_{tn}$ as a result of the noise.

Particle Filter

4. *Correction* (Figure 16.26): Use the current observation $\mathbf{z}_t$ and knowledge of the observation probabilities to set

$$\pi_{tn} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_{tn}) .$$

5. Iterate N times from (2).

6. Normalize $\{\pi_{ti}\}$ so that $\sum_i \pi_{ti} = 1$.

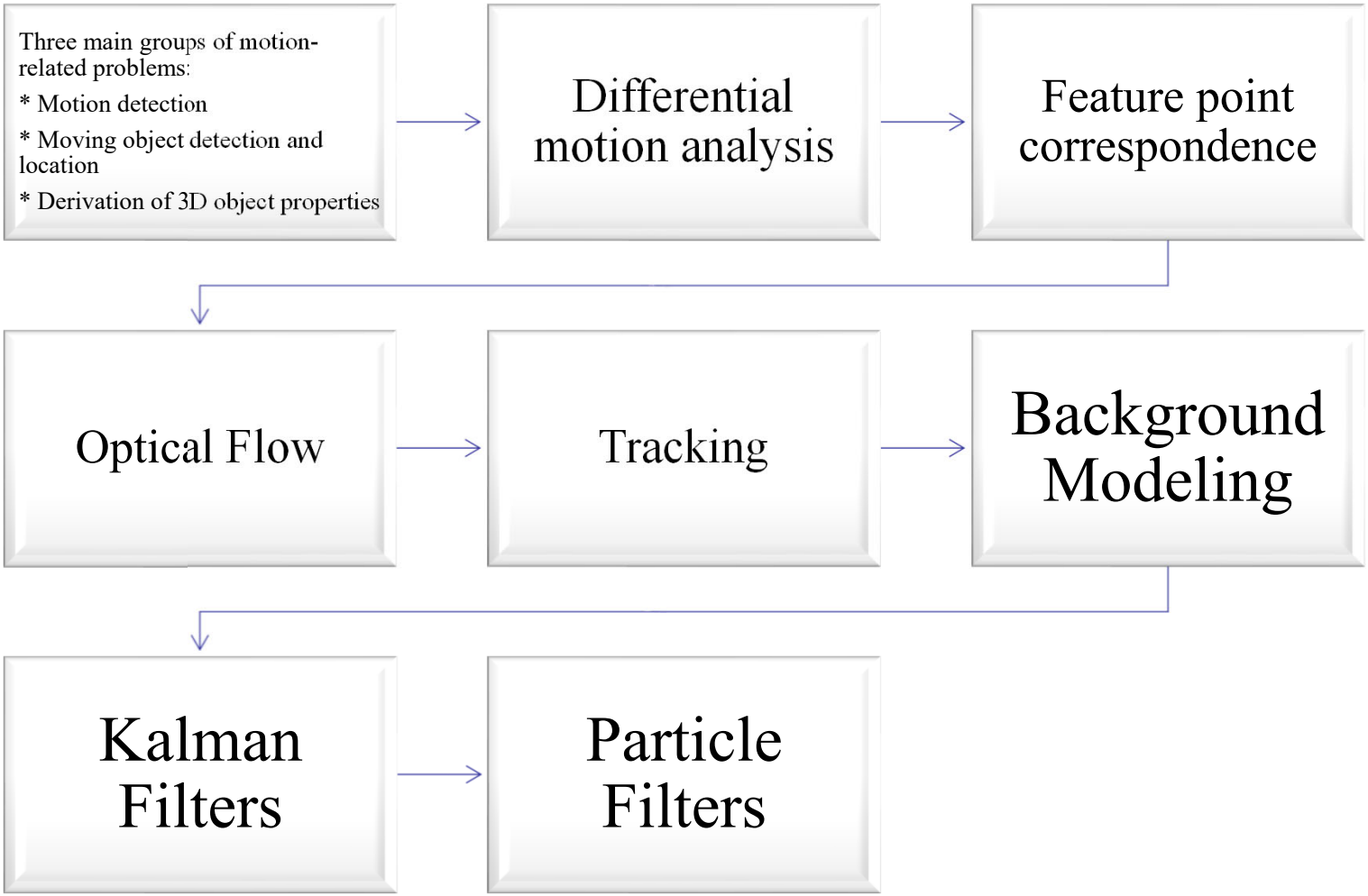
7. Our best estimate of $\mathbf{x}_t$ will now be

$$\mathbf{x}_t = \sum_{i=1}^{N} \pi_{ti} \, \mathbf{s}_{ti} \qquad (16.97)$$

or, more generally for any moment

$$E[f(\mathbf{x}_t)] = \sum_{i=1}^{N} \pi_{ti} \, f(\mathbf{s}_{ti}) .$$

Conclusion


Three main groups of motion-related problems:
* Motion detection
* Moving object detection and location
* Derivation of 3D object properties
Differential motion analysis
Feature point correspondence
Optical Flow
Tracking
Background Modeling
Kalman Filters
Particle Filters

References

- Main:
  - Milan Sonka, Václav Hlavác, Roger Boyle, Image processing, Analysis, and Machine Vision, Cengage Learning, 4th ed., 2015. (Ch 16)
- Further Reading:
  - Fei Chen et al., Visual object tracking: A survey, Computer Vision and Image Understanding, V 222, 2022.
  - Zahra Soleimanitaleb et al., Single Object Tracking: A Survey of Methods, Datasets, and Evaluation Metrics, https://arxiv.org/abs/2201.13066

That's It …