UNIVERSITÄT BASEL

Computer Science / 15731-01 / 2022

Multimedia Retrieval

Chapter 6: Video Retrieval

Dr. Roger Weber, roger.weber@gmail.com

6.1 Introduction

- Video retrieval is a combination of image, audio, and text (subtitles) retrieval. But beyond these basic stream related capabilities, there are a number of additional concepts which we consider in this section:
  - Segmentation (shot detection, scenes)
  - Motion Detection

  There are many more topics in video retrieval but we limit this section to the topics above.
- Video segments are modeled at four different levels:
  - **Frame:** an individual image in the video defining the shortest visual change rate (e.g., 25 frames per second). Although the audio channel has a much higher resolution, the visual channel is often used as finest granularity.
  - **Shot:** a set of frames recorded in a single shooting (without stopping camera but including camera and object movements). May last a few seconds up to several minutes or even hours. A shot encompasses all image, audio and subtitle information and often is the smallest unit for search (frames are often too fine granular for that purpose)
  - **Scene:** a set of shots that share common semantics. In a movie, this could be a discussion between two story characters with alternating viewpoints (the shots) depending on who is talking. A scene is often coherent and consistent in terms of time and location.
  - **Episode:** a set of scenes forming the episode. A movie has often just a single episode, a series may consist of dozens or even hundreds of episodes. Episode segmentation is often at the physical distribution layer (i.e., different files or disks or media); but an episode can also span across several physical carriers.

6.2 Shot Detection

- A shot consists of frames from a single camera shooting. Shot boundaries change the perspective within a scene, or change time and location of the setting if they also mark a new scene. A common characteristic is the rapid change of image information depending on how the shot transition is rendered:
  - **Hard cuts:** there is no cross-over between two subsequent shots and a clear (hard) delineation between the last frame and the first frame of the shots. A hard cut is marked by an abrupt change in the image stream.
  - **Soft cuts:** the two shots are intertwined with each other changing from one shot to the other over the course of multiple frames. Fade in/out, swipes, and other visual effects mark the change of two shots. In contrast to hard cuts, there is no frame that marks the end or the start but a sequence of shared frames for the visual transition effect.

  Hard cuts are often used for camera changes within the same scene like in a discussion between two people changing the viewpoint from one speaker to the other. Soft cuts often occur to visually mark the end of a scene and to direct the attention of the viewer to time and/or location change.
- Indicators of shot boundaries can be found in the video stream. An encoder uses an I-frame if the changes between subsequent frames is too large for a differential (prediction) approach. However, I-frames are also frequently used to allow for quick navigation within the video and occur at frequent intervals. They are hence not often useful for shot detection but can help to reduce some of the efforts.

- **Shot Detection (hard cuts)**
  - A hard cut is an abrupt change of the image stream. In principal, we need a similarity function between two subsequent frames and a threshold that lets us detect a shot (if similarity is below that threshold). Often, we compute distance between subsequent frames rather than similarity values. In this case, a threshold is needed such that distances larger than the threshold indicate a shot boundary.
    - **Pixel based comparison:** a naïve approach is to consider the changes per pixel along the time scale and compute a distance between subsequent frames $\boldsymbol{f}(x,y,i)$ and $\boldsymbol{f}(x,y,i+1)$ as follows ($\boldsymbol{f}()$ is vector function returning red, green, blue channels):

$$d_{naive}(i) = \sum_{x,y} |\boldsymbol{f}(x,y,i) - \boldsymbol{f}(x,y,i-1)|$$

      The problem with this approach is that it is not very robust against camera movements and object movements. A small shift of the camera may lead to very large distances.
    - **Histogram / Moments Comparison:** to have (small) translation, rotation, and scale invariance, moment and histogram features are better suited. In addition, we often consider only the luminance values as frames from two different shots have quite different luminance distributions. The standard approach is histogram over luminance values. Let $\boldsymbol{h}(i)$ denote the histogram (or feature vector) for a frame. We then obtain a better distance measure (we can either use Manhattan distance or a quadratic function):

$$d_M(i) = |\boldsymbol{h}(i) - \boldsymbol{h}(i-1)| \qquad d_Q(i) = \boldsymbol{h}(i)^\top \mathbf{A}\boldsymbol{h}(i-1)$$

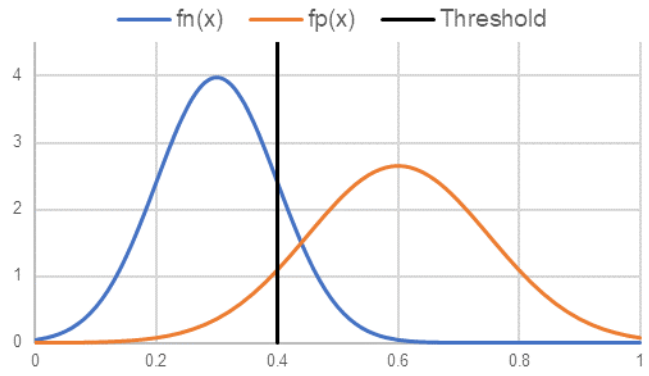
- To learn the best threshold, we already considered the ROC curve in chapter 1 as an excellent tool. Let $f_n(x)$ denote the distribution of distances between two frames belonging to the same shot, and $f_p(x)$ denote the distribution of distances between two frames from different shots. A threshold T is defined such that:
  - $d_m(i) < T$ denotes that frame i belongs to the same shot (no shot boundary; negative case)
  - $d_m(i) \geq T$ denotes that frame i belongs to a new shot (shot boundary; positive case)

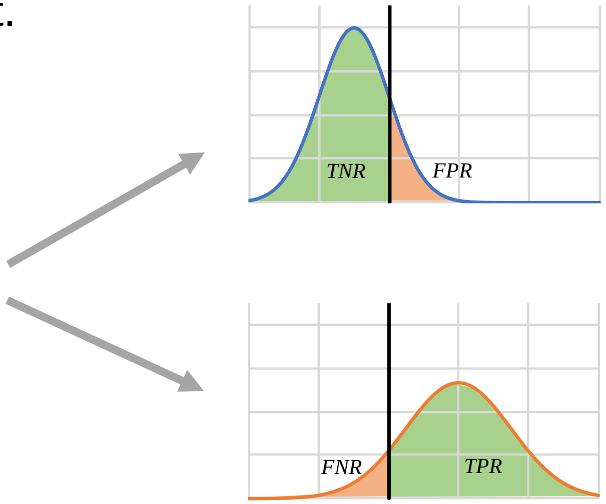
  We now can compute the false/true positive/negative rates as defined in chapter 1:

$$TPR(T) = \int_T^\infty f_p(x)\,dx \qquad FNR(T) = \int_{-\infty}^T f_p(x)\,dx$$

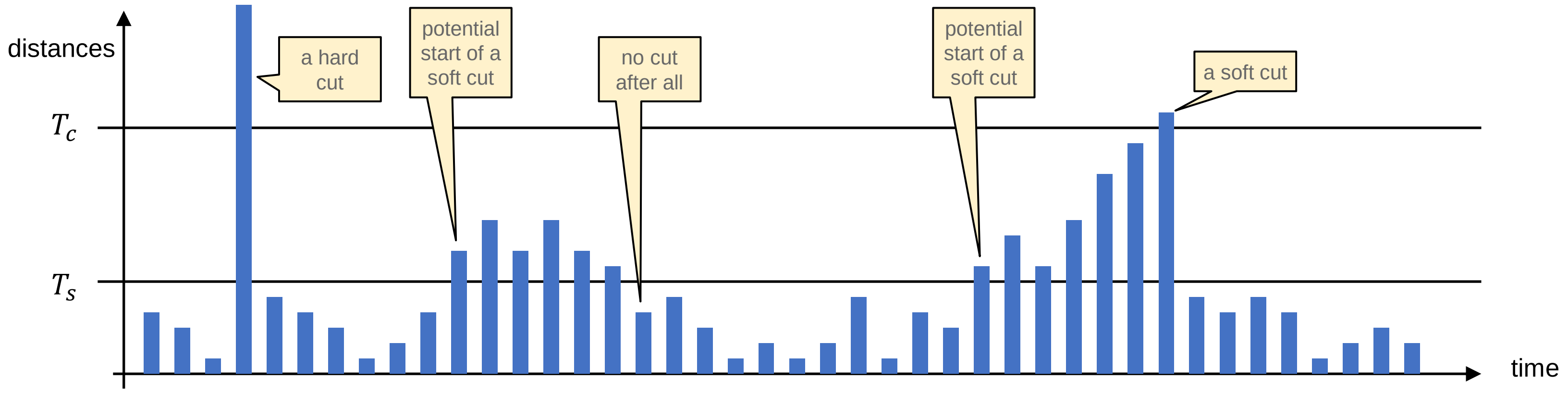
$$TNR(T) = \int_{-\infty}^T f_n(x)\,dx \qquad FPR(T) = \int_T^\infty f_n(x)\,dx$$

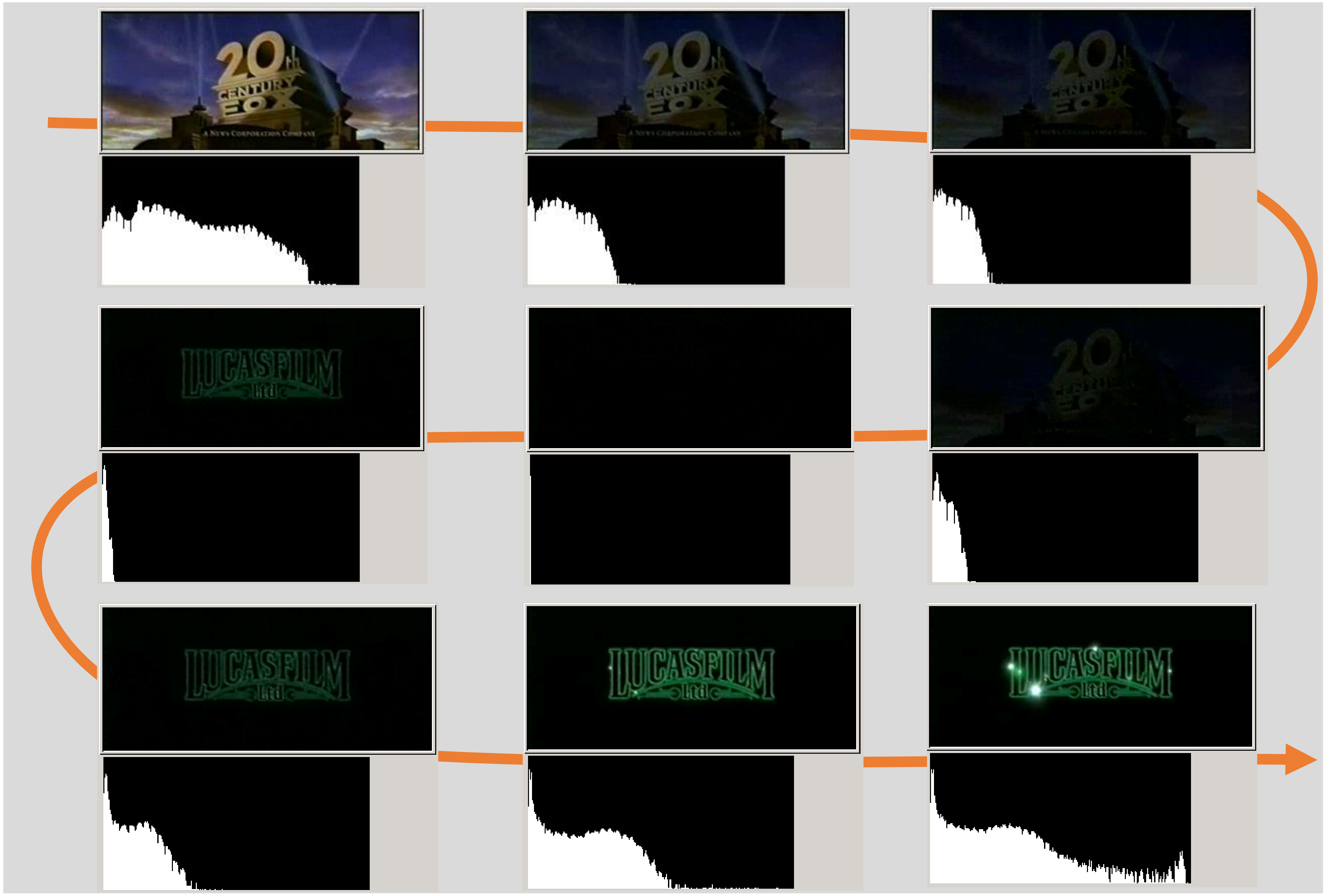
  The best threshold T depends on our objective function, but typically we would select T such that accuracy is the highest. The ROC table provides a simple tool to compute T.





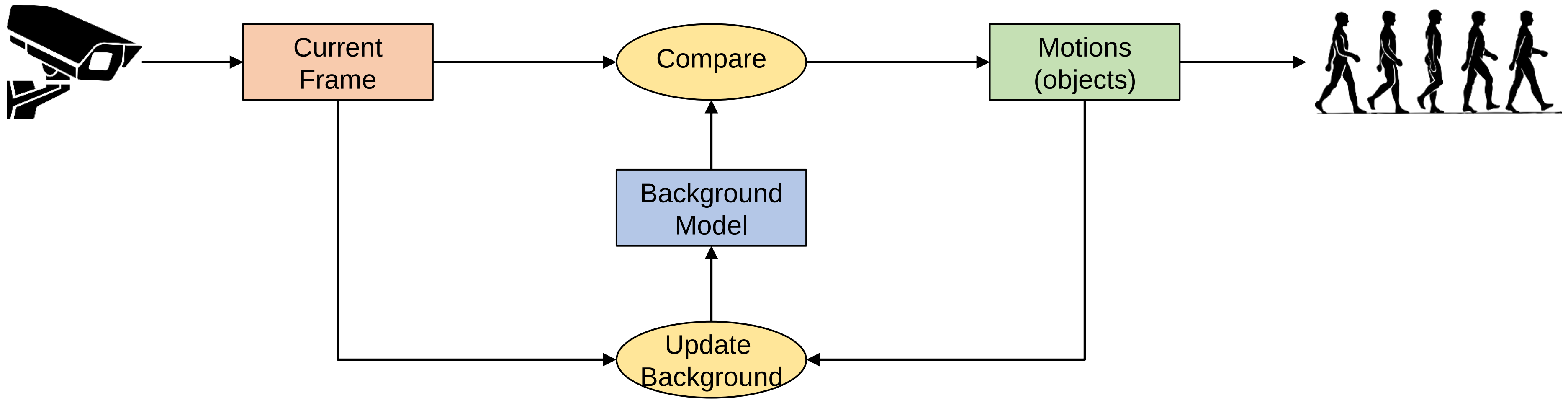
- **Shot Detection (soft cuts)**
  - The detection method above works well for hard cuts but it struggles with visual effects between shots. For instance, a slow fade-out, fade-in effect does not change subsequent images enough to trigger the threshold; but after some time, the image has changed significantly.
  - A first alternative is to model the different transition effects. A fade-out, fade-in is simple to detect (all black screen). Swipes (horizontal or vertical) are splitting the image into two parts (one part from the old shot, one part for the new shot) and gradually change the ratio between the parts. But it takes a lot of coding to model all the visual effects, and new effects can not be detected.
  - **Twin Thresholding** is a generic approach to identify visual transitions from one shot to another. It works with two thresholds: threshold T_c detects (hard) changes between two frames similar to hard cuts. A new threshold T_s is much lower and more sensitive. If the difference exceeds this threshold, it marks the potential begin of a soft cut. The current frame is kept as the reference image and we keep this reference for the next frames until a) the difference to the reference frame exceeds T_c (soft cut detected), or b) the difference falls below T_s again (no cut after all). In both cases, we release the reference frame and use the current frame as a new reference.



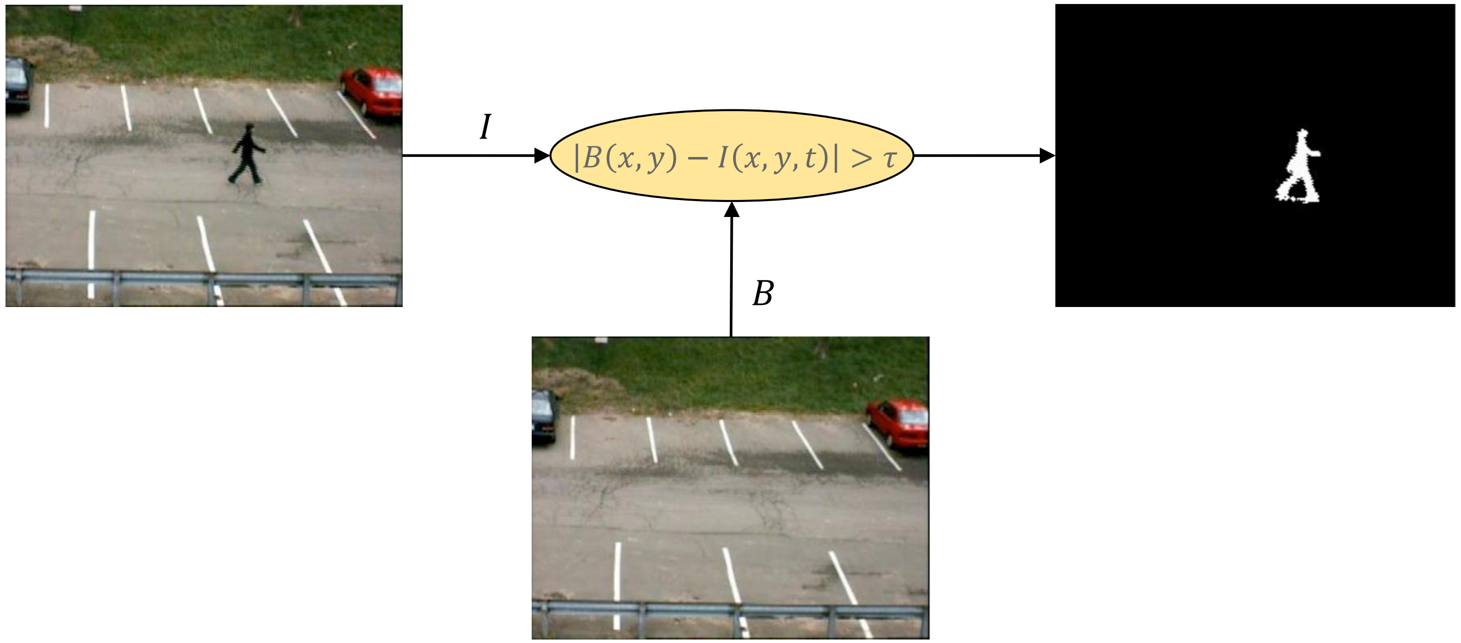


6.3 Motion Detection

- Motion detection has several use cases:
  - Motion compensation in video encoding helps to encode frames with previous frames and the relative motion of blocks. This reduces the number of bits required to encode a frame
  - Surveillance cameras detect and track motion of objects. Often cameras are stationary are objects move in front of the camera.
  - Optical flows analyze relative movements of camera (observer) and objects in the scene. In the area of robotics, optical flows allow to estimate movements and the 3D structure of a scene.
- **Detecting Moving Objects:** The basic assumption is that the camera is stationary and moving objects are the important pieces. We want to identify movements (to trigger an alarm) and the motion vectors of these objects (to track them). The model is fairly straightforward



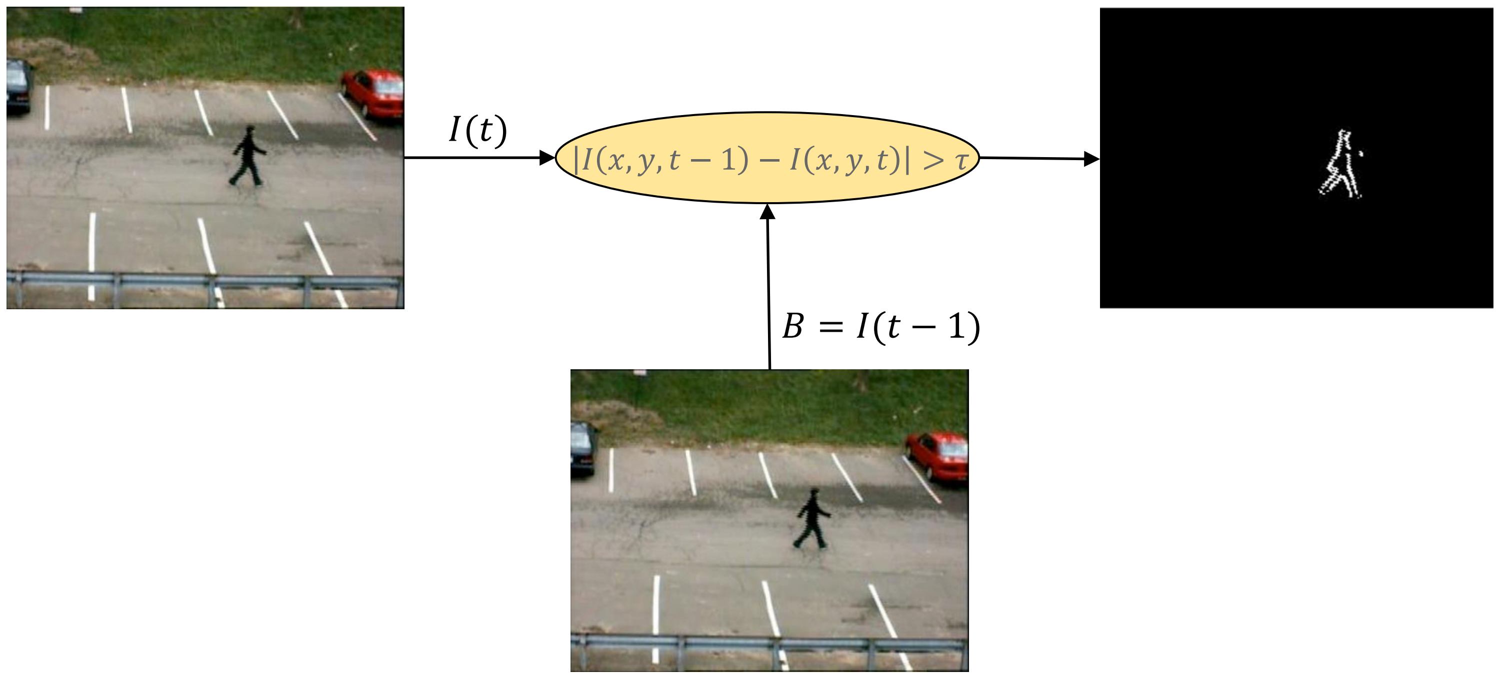
- **Simple Background Subtraction:** the background image is a static, arbitrarily selected image. We assume that the background image does not contain any moveable objects. Pixels are label with “white” if they belong to an object moving, or “black” if they are background.



- Good starting point to extract shape of an object. However, sensitive to illumination changes (weather, position of sun). If the background image changes over time (permanent change of scene), a negative ghost image remains.
- Very sensitive to any movement in the picture even if unimportant (for instance, leaves of a tree moving with the wind, reflections of sunlight)
- Only works if the camera is absolutely static (also no zoom or tilt).

Images by Robert Collins

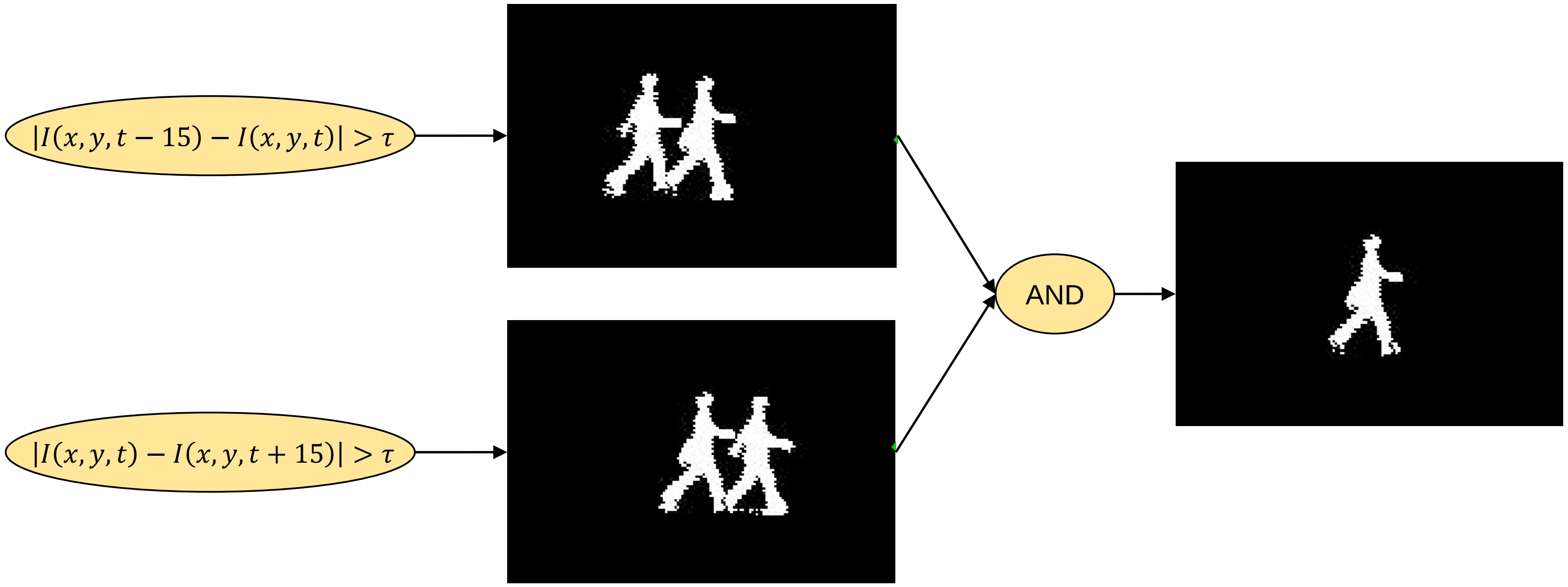
– **Simple Frame Differencing:** instead of a static background image, we build differences between subsequent frames. This allows to adjust to changes over time.



- Robust to scene changes over time; very quick to adapt to lightning changes or even camera motion (incl. zoom and tilt).
- Objects that stop are no longer recognized. If they start again, they leave a negative ghost
- Only changes in the direction of movements are detected. If an edge of an object moves has the same orientation, that edge is not visible. As seen above, only a partial silhouette is captured: the front and the back, but not the top and the bottom part.

Images by Robert Collins

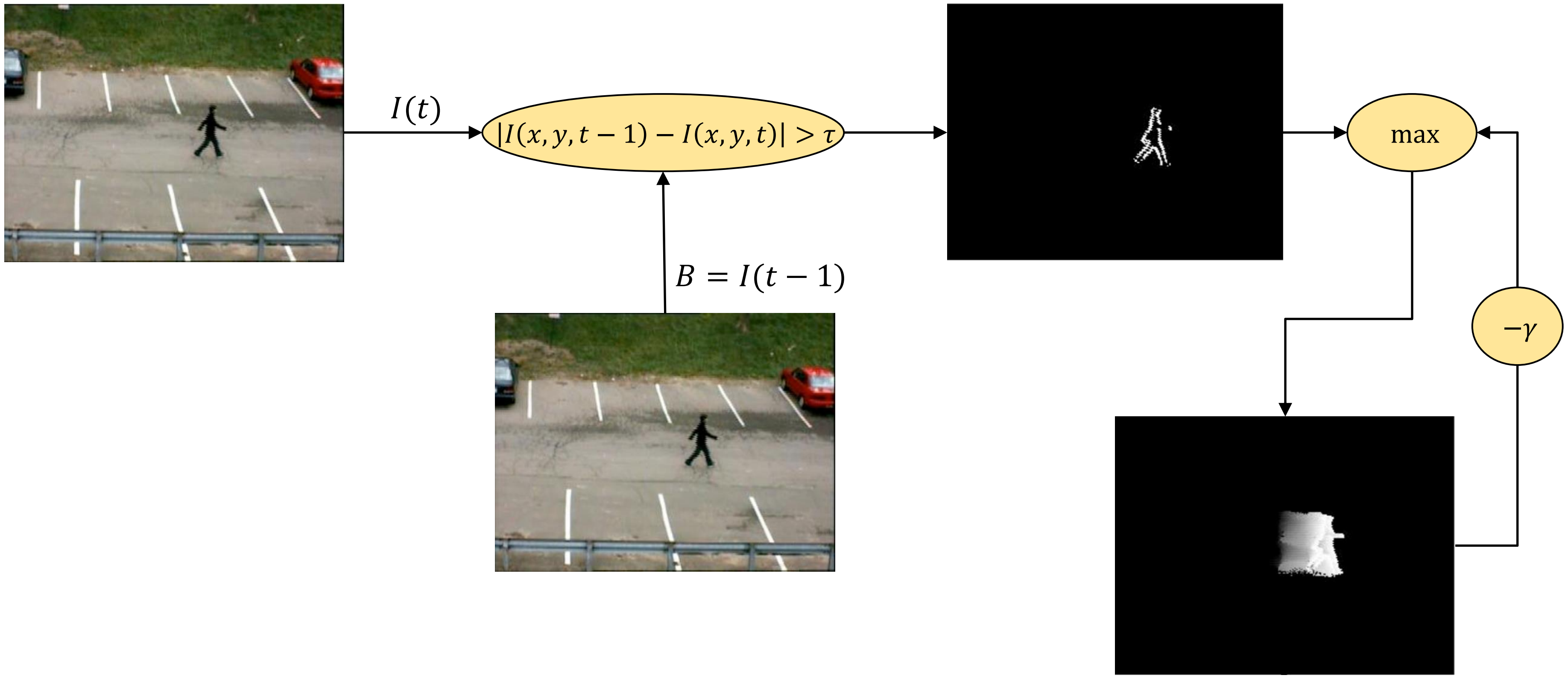
– **Three Frame Differencing:** with the simple frame differencing, we compared subsequent frames. If we enlarge the temporal distance between two frames to compare, we find more complete silhouettes but also two copies of the objects (its starting point and its current point). To eliminate copies, the three frame differencing compares with a frame in the past (say $t-15$) and a frame in the future (say $t+15$). The intersection of the two images leads to the current location of the moving object. Note: this means a delay in the identification of the objects current position.



- Choice of good frame-rate and temporal distance between images depend on size and speed of objects. With the example above, the current position is the intersection of the two difference images (one for the past, one for the future).

Images by Robert Collins

– **Motion History Images:** we compute differences between subsequent images to obtain motion images which are then combined with a linear decay over time. The motion history images provide an impression from where the object is coming.

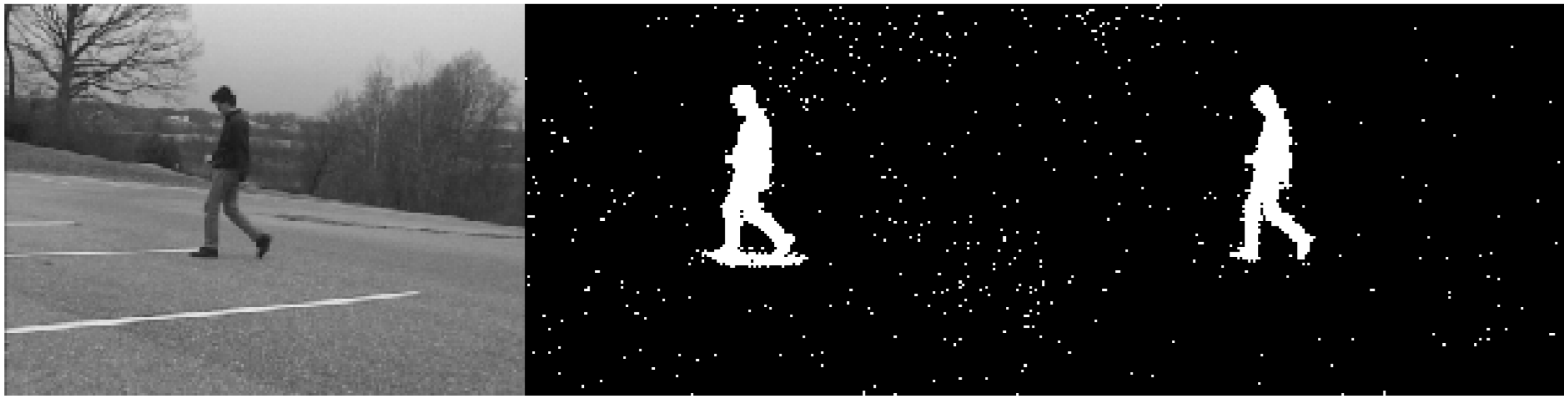


- We obtain the current motion histogram from the previous one by subtracting a chosen value γ (negative values are zeroed) and combining it (max function) with the current motion image. The decay parameter γ adds gray values to the motion history denoting how long ago the motion was detected (the darker the earlier). The larger γ is, the shorter the history of object motions. The motion history images summarizes how much motion occurred over a given time period. We can use it to summarize motion aspects into a feature vector (using histograms or moments like for other features).

Images by Robert Collins

- **Shadow Elimination:** the pixelwise difference methods detect moving objects together with the shadows they cast (see example below). This leads to poor localization, the impression of additional motion, and hence should be avoided. We can distinguish a shadow from the actual object by comparing the color chromaticity of the pixels. A shadow will only change illumination of the background color but chromaticity remains similar. We can adjust the methods with and improved differencing method of two frames:
  - Instead of building differences between pixels of two images, we first map the images to a chromaticity sub-space (e.g., a*b* or HS) ignoring the luminance aspects. The thresholding eliminates shadows but keeps objects (if they are sufficiently different in terms of color than the background).

differencing method: $|B_{chroma}(x,y) - I_{chroma}(x,y,t)| > \tau$



- **Optical Flow:** the previous methods only work for stationary cameras (also no zoom or tilt). To detect motion in arbitrary videos, other methods are required. The most prominent approach is known as the **Lucas-Kanade algorithm.**
  - The basic assumption is brightness constancy, i.e., no abrupt changes in brightness of a pixel across subsequent frames. We are considering a pixel and its motion path over time. Let $I(x(t), y(t), t)$ denote the brightness of a pixel with its path $[x(t), y(t)]$ as a function over time. Brightness constancy than means:

$$I(x(t), y(t), t) = const \quad \text{for small changes of } t$$

  - Let us track the pixel from the frame at time t to the subsequent frame at $t + \Delta\text{t}$ with Δt the time difference between two frames (0.04s with 25 frames per second). The pixel has moved to a new location $x(t+\Delta t) = x(t) + u$ and $y(t + \Delta t) = y(t) + v$. We us a Taylor expansion in the following equation to linearize the equation and to solve for u and v:

$$I(x(t), y(t), t) = I(x(t + \Delta t), y(t + \Delta t), t + \Delta t) = I(x(t) + u, y(t) + v, t + \Delta t)$$

$$I(x(t) + u, y(t) + v, t + \Delta t) \approx I(x(t), y(t), t) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}\Delta t$$

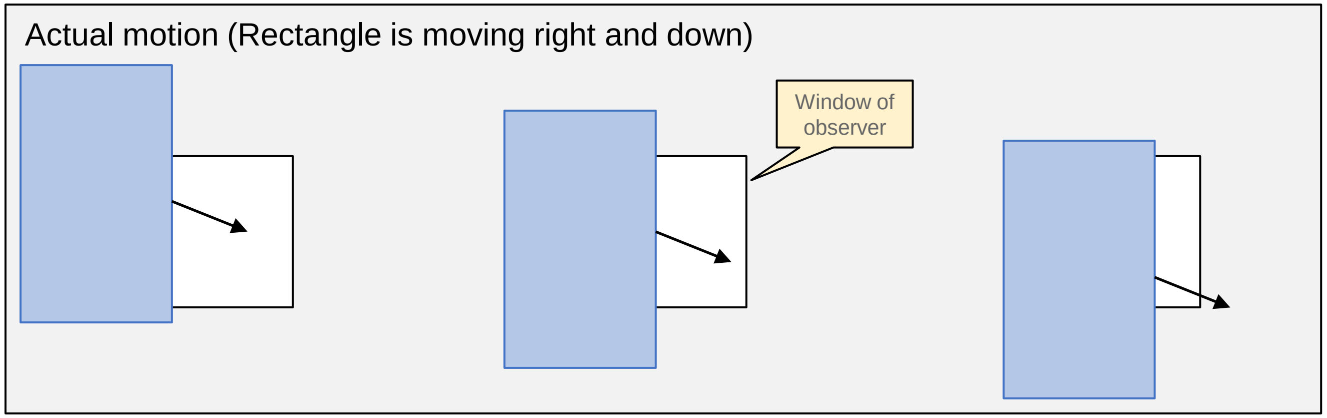
$$0 \approx \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}\Delta t$$

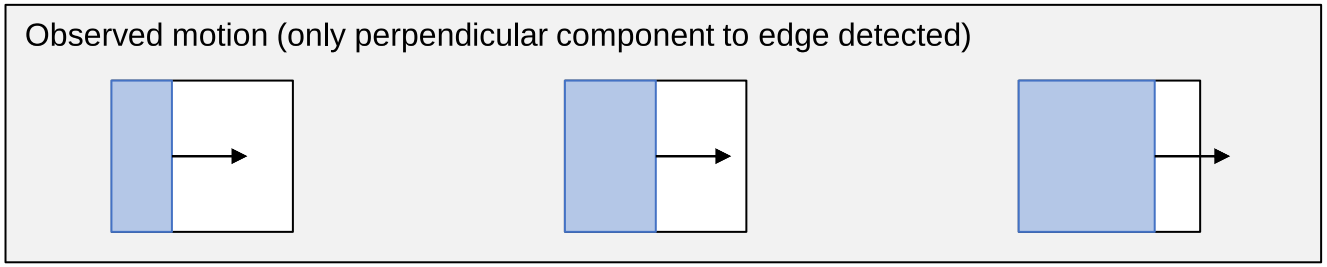
  - The partial derivatives are the brightness gradients in x, y and t dimension. We can compute $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ with a Sobel operator to obtain $I_x(x, y)$ and $I_y(x, y)$ at time t. The term $\frac{\partial I}{\partial t}\Delta t$ is the difference $I_t(x, y)$ of brightness between subsequent frames at time t and $t + \Delta\text{t}$.

– We obtain the final equation for our motion estimate:

$$I_x(x,y) \cdot u + I_y(x,y) \cdot v = -I_t(x,y) \qquad \text{at time } t$$

Given that we can observe the partial derivatives I_x, I_y and I_t, we have to solve the above linear equation for u and v to obtain the motion vector for the pixel at (x,y) and time t. As we see, we have only one equation but two unknowns. Hence, there are many possible solutions. If (u,v) is a solution, then $(u+u', v+v')$ is a solution if (u',v') is perpendicular to $\big(I_x(x,y), I_y(x,y)\big)$. In other words, we can not measure the motion along an edge but only perpendicular to edges. This is known as the **aperture problem**:




Observed motion (only perpendicular component to edge detected)

Other Examples:
https://en.wikipedia.org/wiki/Motion_perception

http://farm5.static.flickr.com/4044/4172972319_7c070bdcbb_o.gif

- Lucas-Kanade solved the aperture problem by considering a 5x5 window around the current pixel assuming that motion in such a small window is approximately the same. This then leads to 25 equations for the two unknowns:

$$I_x(x+a,y+b)\cdot u + I_y(x+a,y+b)\cdot v = -I_t(x+a,y+b) \quad \text{for all: } -2 \le a,b \le 2$$

$$\underbrace{\begin{bmatrix} I_x(x-2,y-2) & I_y(x-2,y-2) \\ \vdots & \vdots \\ I_x(x,y) & I_y(x,y) \\ \vdots & \vdots \\ I_x(x+2,y+2) & I_y(x+2,y+2) \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\boldsymbol{d}} = \underbrace{\begin{bmatrix} -I_t(x-2,y-2) \\ \vdots \\ -I_t(x,y) \\ \vdots \\ -I_t(x+2,y+2) \end{bmatrix}}_{\boldsymbol{b}}$$

$$\mathbf{A}\quad \boldsymbol{d} \;=\; \boldsymbol{b}$$

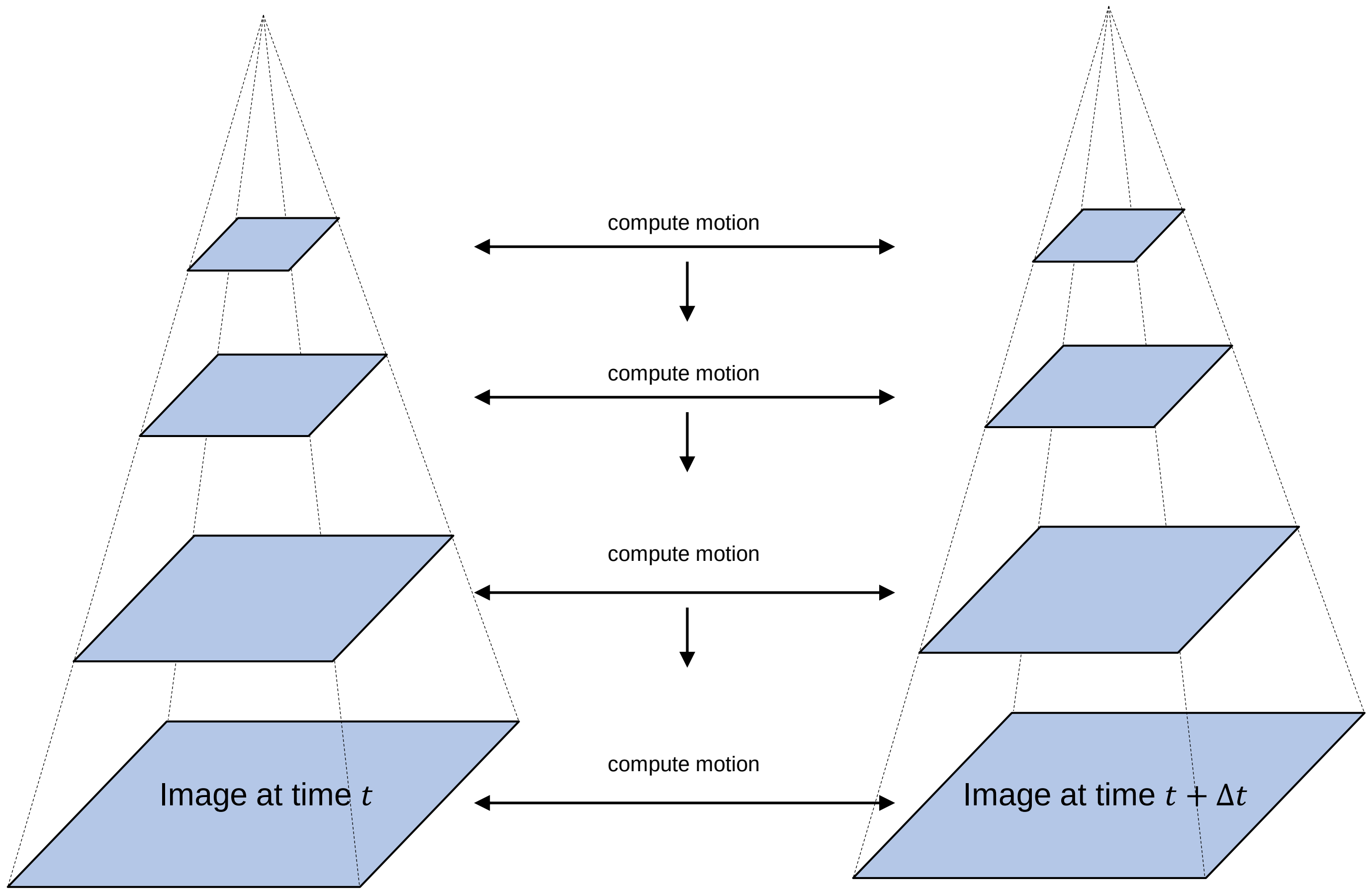
- Since there are more equations than unknowns, there is no exact answer. Instead, we minimize $||\mathbf{A}\boldsymbol{d} - \boldsymbol{b}||^2$ by solving its gradient for zero which leads to

$$(\mathbf{A}^\top\mathbf{A}) \quad \boldsymbol{d} \; = \; \mathbf{A}^\top\boldsymbol{b}$$

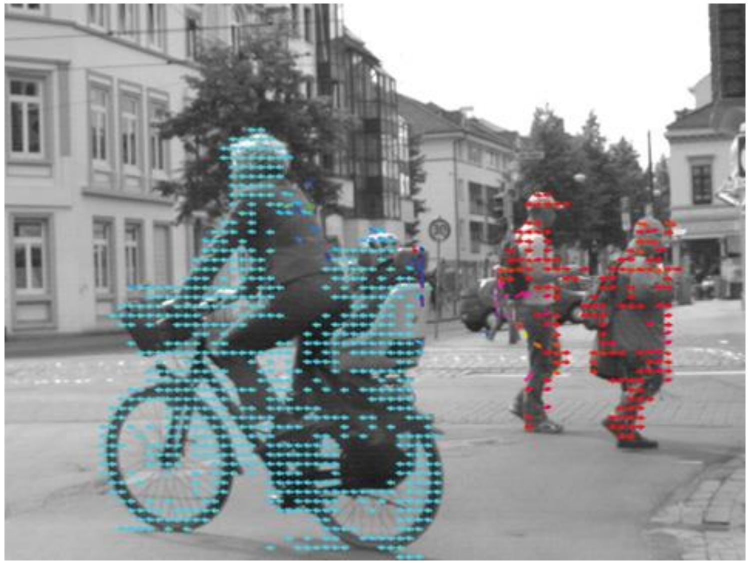
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

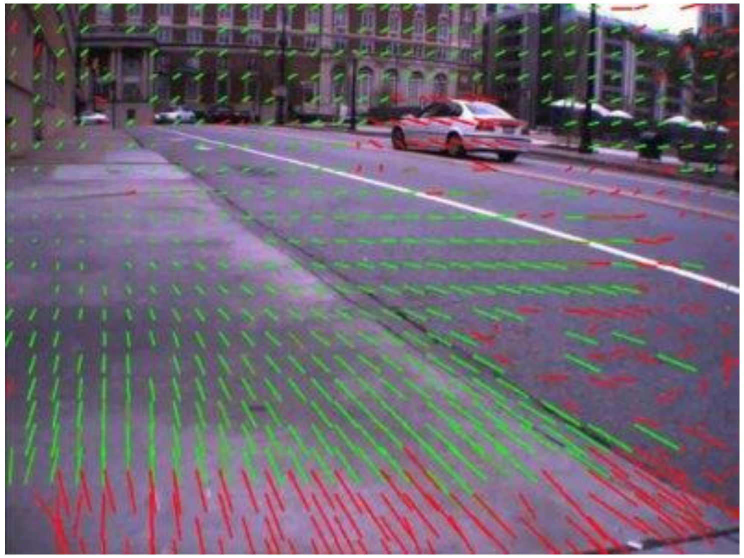
  The summations are over all pixels in the 5x5 window around the current point. To solve the equation, $\mathbf{A}^\top\mathbf{A}$ should be invertible, should not have too small eigenvalues, and the ratio between the two eigenvalues should not be too large. Hence, this works best for corner points (or key points or Harris points) but not for edge points and for points in the flat.

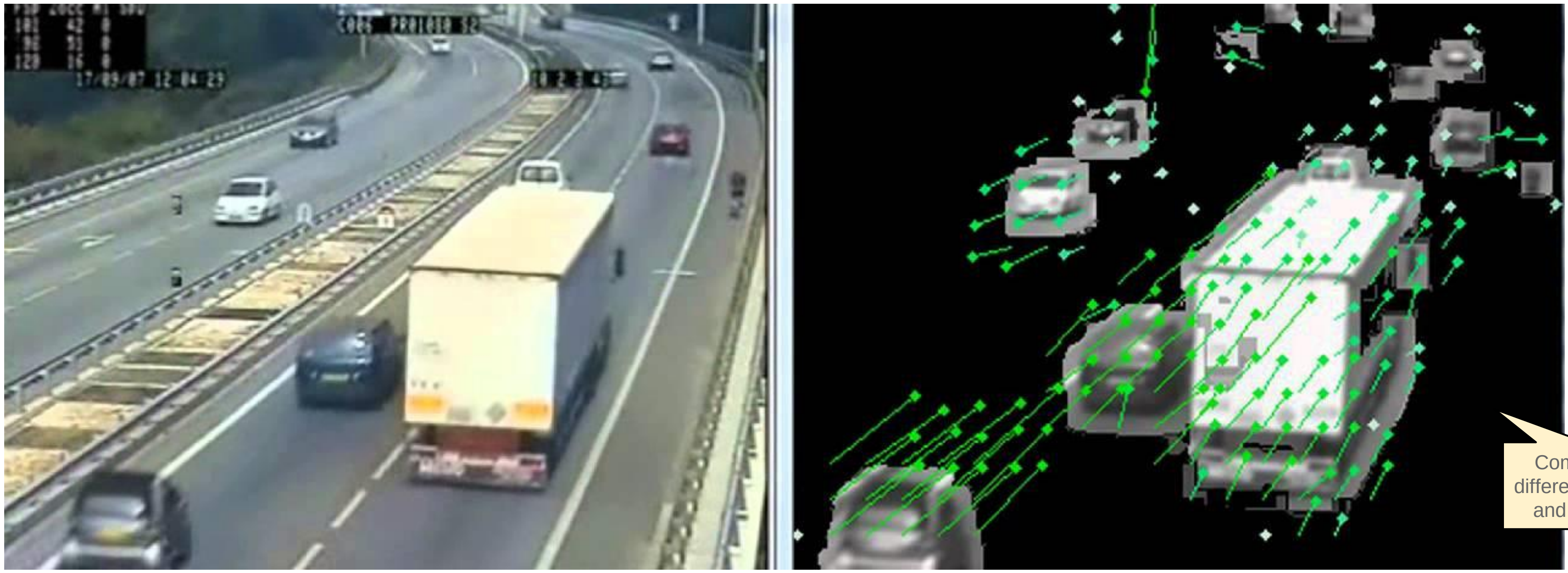
– The basic method works only well for small displacements. For large displacements, we can use a Gaussian pyramid of the image and estimate flows at each scale:



– Optical Flow: Examples (various sources)







Combination of differencing method and optical flow

6.4 Literature and Links

- Wikipedia, Background subtraction, retrieved 2017 from https://en.wikipedia.org/wiki/Background_subtraction
- B. D. Lucas and T. Kanade (1981), An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121—130
- Frameworks and Libraries
  - **scikit-video** (http://www.scikit-video.org) is a Python library for video processing
- Interesting courses at other universities
  - Multimedia Content Analysis,National Chung Cheng University, Taiwan, https://www.cs.ccu.edu.tw/~wtchu/courses/2014f_MCA/lectures.html#00
  - Computer Vision, University of Washington, USA, https://courses.cs.washington.edu/courses/cse455/
  - Computer Vision, Penn State University, USA, http://www.cse.psu.edu/~rtc12/CSE486/
  - Computer Vision, University of Illinois, USA, https://courses.engr.illinois.edu/cs543/sp2012/
  - Computational Photography, University of Illinois, USA, https://courses.engr.illinois.edu/cs498dh/fa2011/