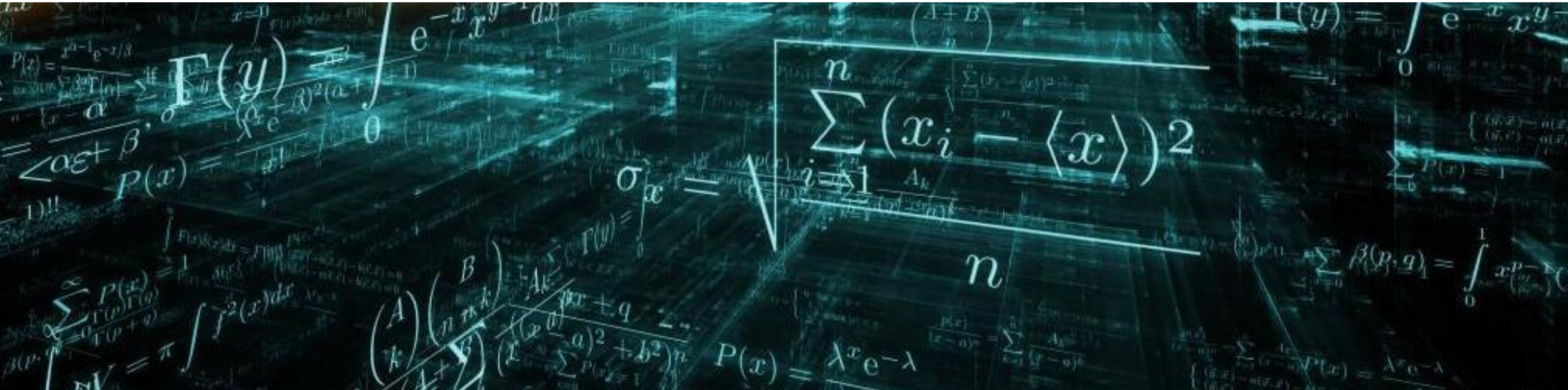
Universität Basel

Multimedia Retrieval

Chapter 13: Spatiotemperal Features

13.1 Introduction
13.2 Shot Detection
13.3 Motion Detection
13.4 Literature and Links

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



13.1 Introduction

- Video retrieval is the task of finding and accessing a video's visual frames, audio tracks, and text elements such as subtitles. These different media types exist together along a time line, so effective retrieval requires knowing how they are stored, organized, and linked over time. Beyond these basic skills, this section introduces techniques for splitting a video into meaningful segments and for detecting and using motion in analysis. The field includes many specialized topics, but the focus here remains on the core concepts that form a foundation for further study.
- Think of a video as made of layers. Inside, the content is arranged at different levels of detail that make the video easier to navigate. Segmentation is the process of splitting the video into those levels. It lets algorithms jump to specific moments, index content efficiently, and analyze large collections without treating every second the same.
  - The smallest, most detailed unit is the **frame**. A frame is a single still image captured at one moment. A video is a sequence of these frames. Depending on the frame rate, each second of video contains dozens of separate visual states. Audio is sampled much faster, but frames are the basic unit for visual inspection. In practice, frames are rarely used directly for search because there are too many and they contain too fine granular details. They remain essential for object recognition, visual tracking, and annotation.
  - A **shot** sits above the frame. It is a continuous sequence of frames recorded without interruption. A shot captures motion in front of the camera and changes in viewpoint caused by the camera's movement. It can be short or very long, depending on artistic intent or recording conditions. A shot includes image data, synchronized audio, and any subtitles. Because it is a single coherent unit, the shot often serves as the basic building block for indexing and retrieval systems.
  - Multiple shots are grouped into a **scene**. A scene brings together shots that share the same context. This usually means the same location, the same time, or a continuous action, such as two characters talking with each and the camera view switching between the two persons. In films, scenes often focus on a narrative event, like a conversation filmed from different camera angles. Even if shots change quickly, the scene ties them together by a common setting or storyline.
  - At a broader level there is the **episode**. A film typically equals a single episode, while a series includes many. Episodes are both narrative and practical units. They are usually saved as separate files and may be kept on different storage devices. Yet they maintain continuity within the larger work and are essential for organizing and finding content in long running productions. With movies and series, this level also features rich meta-data content including plot details, keywords, characters, actors, and other useful information for retrieval.

- Video retrieval supports a wide range of practical uses
  - A common example is large media archives. Broadcasters, streaming services, and film studios keep vast collections of footage gathered over many years. Retrieval systems let archivists and editors find specific moments, like a particular interview, a landscape shot, or a recurring character. Instead of watching hours of material, they can pinpoint the exact shot that shows an event. This ability is invaluable for producing documentaries, assembling highlight reels, or making promotional material.
  - Another important use is surveillance and security. Around-the-clock video would overwhelm human reviewers without automated help. Searching by motion patterns, object detection, or scene changes helps investigators find when a person enters a room, a vehicle crosses a boundary, or an object is removed from a scene. Segmentation is especially important because continuous recordings must be split into manageable parts that systems can scan efficiently. The typical shot detection discussed later in this chapter is not appropriate for such settings.
  - In scientific and medical research, video retrieval makes it easier to analyze long or complex recordings. In wildlife studies, researchers may need to find a specific behavior, such as feeding or migration, within hundreds of hours of footage. In medicine, recordings from endoscopies or surgeries can be segmented to mark important moments, allowing physicians to review past cases or train newly qualified staff. Retrieval based on motion or visible change helps pinpoint steps in a process without watching every minute.
  - In entertainment and recommendation systems, video retrieval enables personalized suggestions. By recognizing scenes, settings, and character appearances, platforms can learn what each viewer prefers. This lets them build curated watch lists and highlight moments a viewer may enjoy. Retrieval also supports interactive storytelling, where users explore branching paths by selecting scenes or revisiting key events.
  - Creative professionals use retrieval techniques to manage production assets. Directors, animators, and editors often revisit earlier material to preserve visual continuity, reuse elements, or find specific scenes. A well-segmented archive allows direct access to shots with particular lighting, camera movement, or character positions.
  - Video retrieval plays a growing role in sports coverage because news outlets and broadcasters need fast summaries. Modern sports productions create huge amounts of footage, often from many cameras that record the same action from different angles. Journalists and editors rarely have time to watch all this material, especially when they must produce highlights soon after a match. Retrieval systems find key moments such as goals, fouls, breakaways, and shifts in momentum. They detect typical motion patterns, crowd reactions, and commentary cues to isolate the most newsworthy segments. This helps editors assemble concise summaries that show the game's main events and deliver timely, clear updates to viewers.

- In recent years, advances in cross-modal transformer architectures have greatly sped up progress and renewed interest in video retrieval. These improvements have solved many problems in linking visual, audio, and text modalities and have produced retrieval systems that are more accurate, flexible, and scalable. Here is a look at how and why this has changed the field:
  - The main challenge in video retrieval is combining very different data types: moving images, audio tracks, and subtitles or transcripts, and reasoning about them over time. Traditional systems often processed visual frames separately or combined them roughly, losing temporal and multimodal coherence. Cross-modal transformers are built to fuse these modalities and learn from their interactions in a single embedding space.
  - Because of their attention-based architecture, transformers can model relationships across modalities, for example how visual events relate to spoken words, and across time, for example how a scene develops over several seconds. That makes retrieval systems more powerful. Instead of only finding a single similar frame, they can answer complex queries such as show me the moment when the person says X while doing Y.
- We will not discuss these advanced techniques here because the field is changing quickly and research done today will soon be outdated. These technical improvements have several practical implications:
  - First, better retrieval accuracy makes large video archives, streaming services, surveillance databases, and educational video collections far more searchable and usable. It enables natural language search, for example "When does a character in a movie say X?", instead of relying only on keyword-based metadata.
  - Second, richer cross-modal understanding of vision, audio, and on-screen text enables retrieval in situations where traditional metadata is insufficient, such as user-generated social media content, footage with spoken dialogue, or videos containing embedded text.
  - Third, generative and long-context models make it possible to manage very large, diverse video collections, including long-form videos, without manual labeling or rigid indexing. This lowers costs tremendously.
  - Finally, as models become more powerful and efficient, video retrieval will enable new uses in media editing, content moderation, accessibility such as subtitle-based browsing, archiving, and interactive video tools.
- Despite progress, several challenges remain. Modeling temporal dependencies over very long videos is still hard, and even powerful transformer models require high computing power and memory for long sequences. Combining different modalities can fail when some inputs are noisy or irrelevant, for example background audio, unrelated on-screen text, or abrupt scene cuts. Finally, evaluation is still a problem. The field lacks standard benchmarks that cover the full range of real-world video retrieval tasks.

13.2 Shot Detection

- A shot is a sequence of frames continuously recorded by a camera. A shot boundary can change the viewpoint in a scene or signal a new scene that shifts the time or place. Often there is a sudden change in the image, depending on the type of transition. The audio track often continues without a noticeable change, but it can also change abruptly at the shot boundary.
  - **Hard cuts:** There is no visual overlap between consecutive shots, and the last frame of one shot is clearly distinct from the first frame of the next. A hard cut happens when the image changes suddenly.
  - **Soft cuts:** The two shots blend together, switching from one to the other over several frames. Visual effects such as fades and swipes show the change between them. Unlike hard cuts, there is not a single frame that marks the end or the start. Instead, a series of frames overlap to create the transition.

  Hard cuts are commonly used to switch the camera angle within a scene, such as when two people are talking and the viewpoint shifts from one person to the other. Soft cuts are used to signal the end of a scene and to guide the viewer's attention to a change in time or location.
- You often can tell where one shot ends and another begins by looking at the video encoding. When the difference between frames is too big, the encoder uses an I-frame. These frames also help with fast navigation in the video and show up a lot. They're not great for finding shots, but they do make things a bit easier.
- **Shot Detection (hard cuts)**
  - A hard cut is a sudden change in the video frames. To detect one, we compare two consecutive frames and measure how different they are. We can use a change threshold: if the distance between frames is above that threshold, we mark it as a shot boundary.
    - **Pixel based comparison:** One simple way to analyze the changes in an image over time is to calculate the difference in color for each pixel between two consecutive frames. This can be done by comparing the values of the red, green, and blue channels for each pixel in the two frames. Let $\boldsymbol{f}(x, y, t)$ be a vector function returning red, green, blue values for the pixel at position (x, y) and at time t:

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

- **Shot Detection (hard cuts)**
  - **Histogram / Moments Comparison:** to account for small changes in translation, rotation, and size, it's best to use moment and histogram features. We usually only look at the brightness values because shots typically have distinct brightness distributions that abruptly change from one shot to the next. The usual method is based on histograms of brightness values. Let's use $\boldsymbol{h}(t)$ to represent the histogram (or feature vector) for a frame at time t. Then we can get a more accurate distance measure with:

$$d_M(t) = |\boldsymbol{h}(t) - \boldsymbol{h}(t-1)| \qquad d_H(t) = \boldsymbol{h}(t)^{\mathsf{T}} \cdot \mathbf{A} \cdot \boldsymbol{h}(t-1)$$

- Previously, we introduced the ROC curve to find hyper parameters for classifiers. Let's use $f_n(x)$ to show the distribution of distances between frames in the same shot, and $f_p(x)$ to show the distribution of distances between frames in different shots. We define a threshold as:
  - $d(t) < T$ indicates that the frame t is part of the same shot (no shot change; negative scenario)
  - $d(t) \geq T$ denotes that frame t belongs to a new shot (shot boundary; positive case)

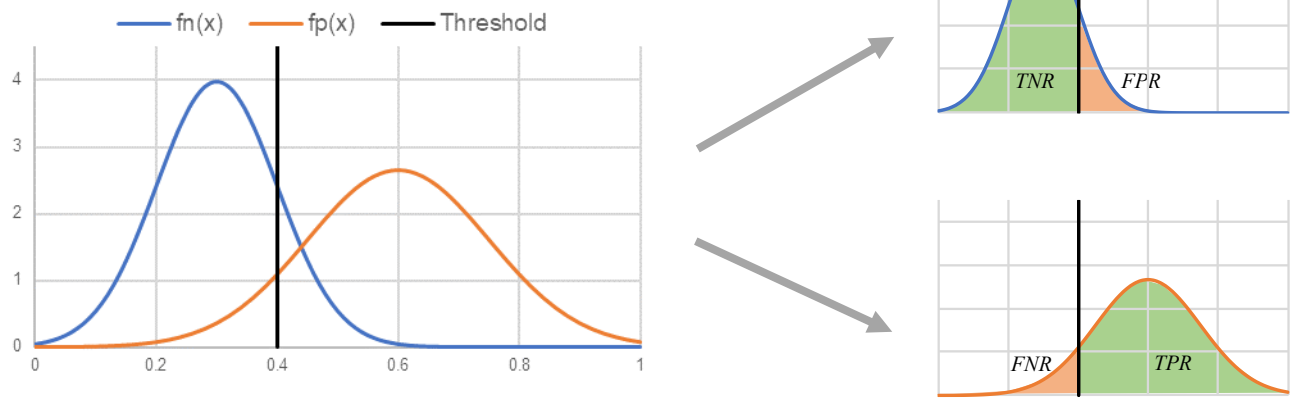
  We can calculate the rates of false positives, true positives, false negatives, and true negatives. The best threshold depends on our objective function, but usually we would select one that maximizes accuracy. The ROC table is a simple tool to compute the threshold T, as discussed in a separate chapter of this class.

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

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

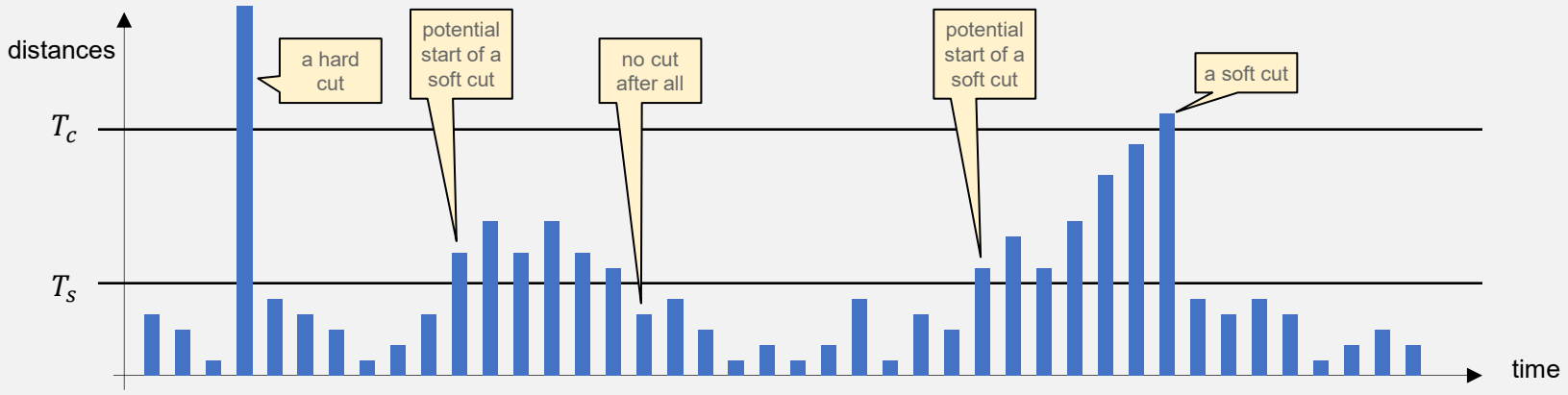
$$FNR(T) = \int_{-\infty}^{T} f_p(x)\,dx$$

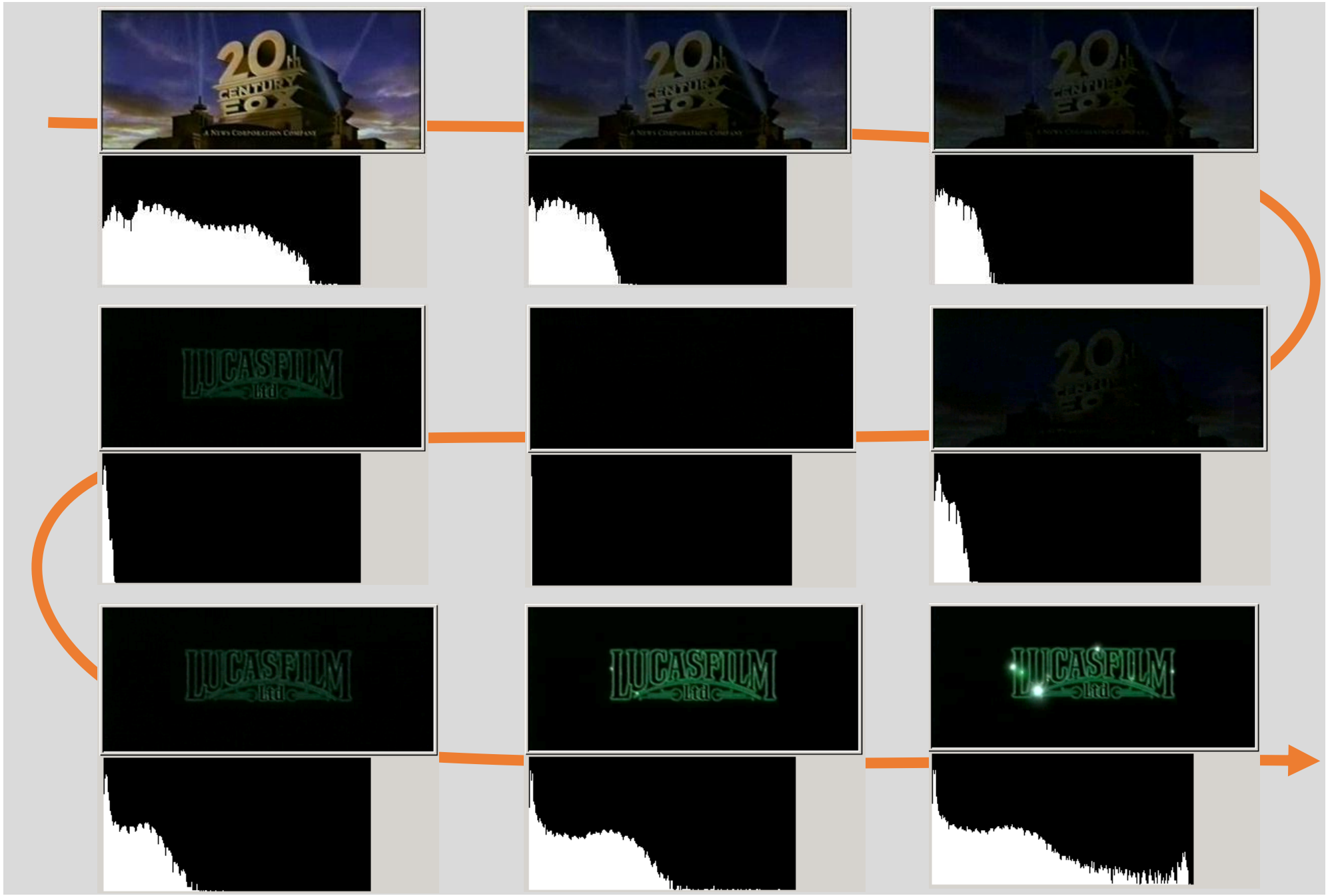
$$FPR(T) = \int_{T}^{\infty} f_n(x)\,dx$$



- **Shot Detection (soft cuts)**
  - The method discussed before is effective for identifying abrupt cuts, but it has difficulties detecting slow visual transitions between shots. For example, a gradual fade-out and fade-in doesn't cause enough change in the brightness histograms to exceed the threshold. However, after some time, the image has changed noticeably.
  - One option is to model various transition effects and to recognize them in the video stream. A fade-out, fade-in is easy to see (the screen turns all black). Swipes (side to side or up and down) divide the image into two parts (one from the old shot, one for the new shot) and slowly change the ratio between them. However, it requires a lot of coding/learning to model all possible visual effects, and new effects cannot be identified.
  - **Twin Thresholding** is a method for finding slow, visual changes from one frame to another. It uses two thresholds: T_c to detect hard cuts as discussed previously; and T_s, a lower and more sensitive threshold, to identify the potential start of a soft cut. When visual changes between subsequent frames exceed T_s, the current frame becomes the reference image, and we keep using it as the reference until either
    a) the difference to the reference frame exceeds the hard cut threshold T_c, or
    b) the difference falls below the threshold T_s again.

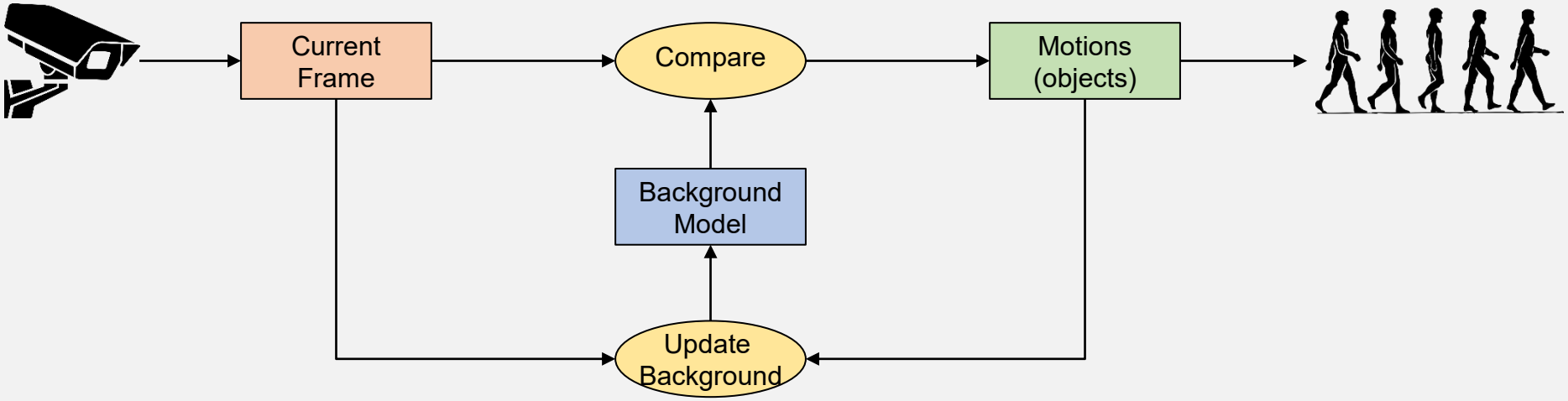
    In both cases, we release the reference frame and continue with the search for the next cut.
  - See the next page for an example of a fade-out/fade-in effect with the changes of the brightness histograms.



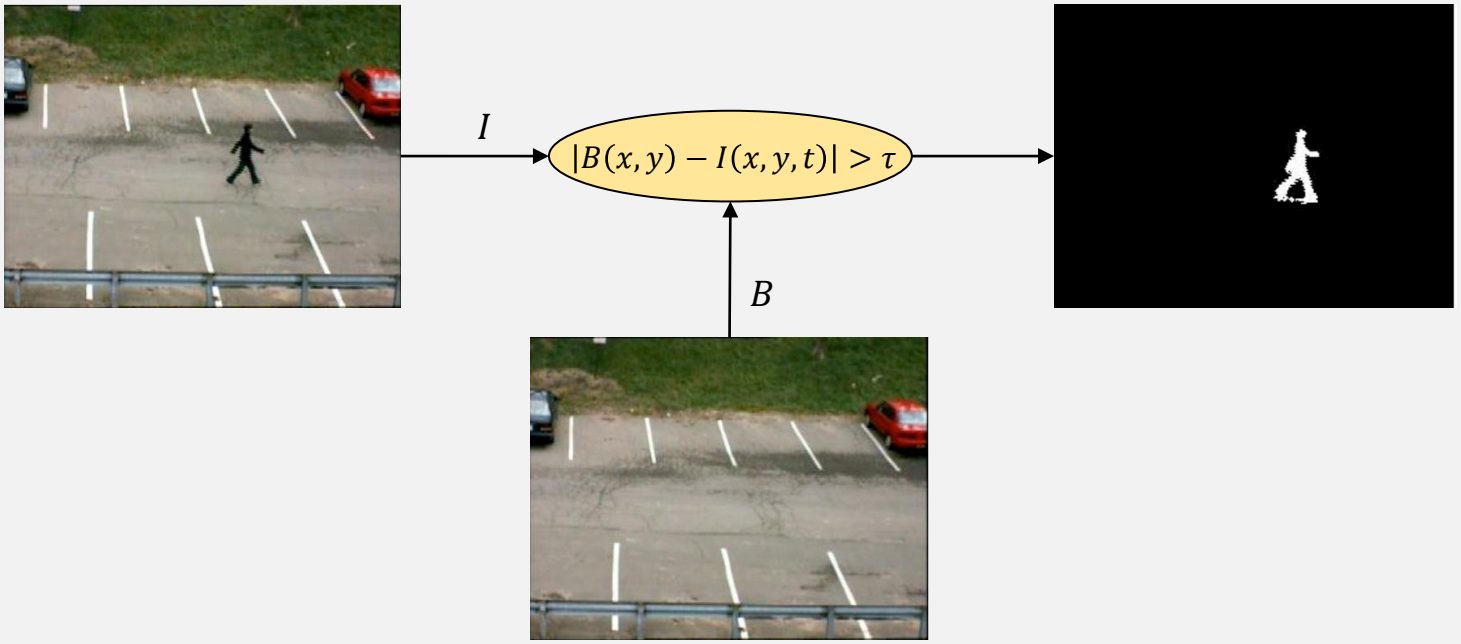

20th CENTURY FOX
A News Corporation Company
LUCASFILM ltd

13.3 Motion Detection

- Motion detection has many practical applications:
  - Video motion encoding involves encoding frames by comparing them to previous frames and considering the movement of blocks. This reduces the amount of data required to encode a frame.
  - Surveillance cameras can detect and track the movement of objects. The cameras remain stationary while objects move in front of them.
  - Optical flows are used to analyze the movements of a camera and objects in a scene. In robotics, they help estimate movements and the 3D structure of a scene.
- **Detecting Moving Objects:** The main idea is that the camera doesn't move, and we focus on objects that are moving. We want to detect movements (to set off an alarm) and track the motion of these objects.



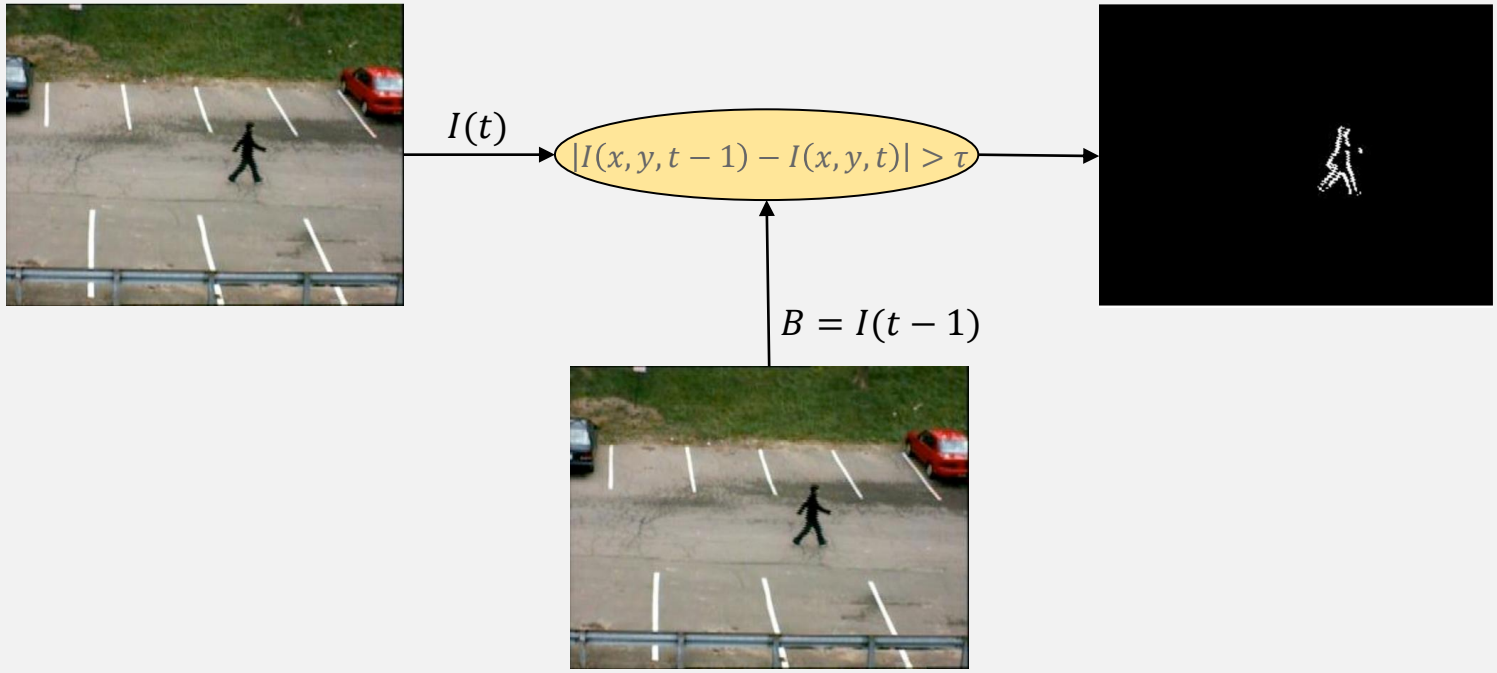
- **Simple Background Subtraction:** The background image is a fixed, initially chosen image. We assume that the background image does not have any objects that can move. To this end, we take the absolute difference between the pixels of the current frame $I(x, y, t)$ and the background pixels $B(x, y)$. Pixels in the difference image are marked as white if they differ between current and background frame (they are part of a moving object), or they are black if there is no difference to the background image.



Images by Robert Collins

This approach offers a useful first step toward identifying the outline of a moving object, yet it comes with clear limitations. Changes in lighting, whether caused by shifting weather or the sun's movement, can distort the result. Alterations in the background also create problems. When an object such as a parked car leaves the scene, the method often retains a faint imprint of it, leaving a ghost image visible across subsequent frames. Minor movements that are irrelevant to the task, like leaves rustling or brief reflections of sunlight, can still trigger the method and introduce noise. Finally, the technique depends on a completely stationary camera. Any zooming, tilting, or panning disrupts the background model and prevents reliable detection.

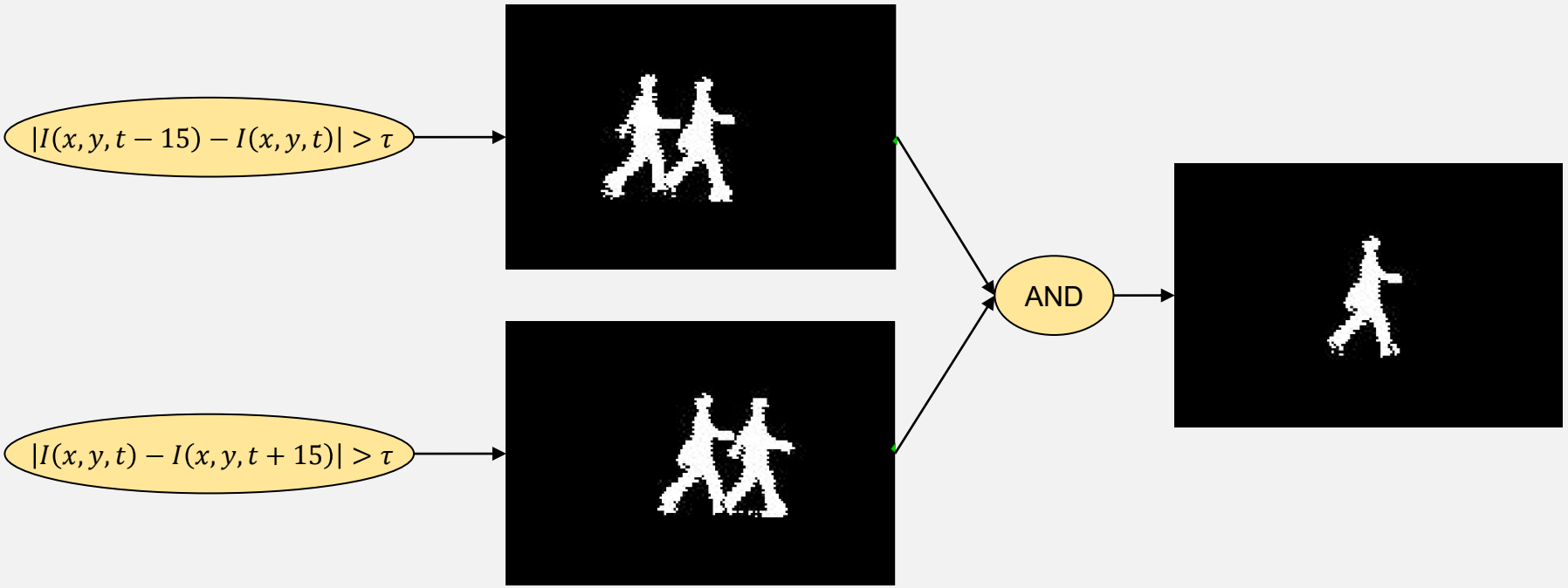
- **Simple Frame Differencing:** Rather than using a still background image, we create variations between consecutive frames. This enables us to adapt to changes as time progresses. We calculate the changes of pixels in subsequent frames and mark pixels white if it changed as before (or black if it did not change):



Images by Robert Collins

This method adapts well to scenes that evolve over time. It can adjust to sudden shifts in lighting and can continue to function even when the camera moves, zooms, or tilts. Despite this flexibility, it has several drawbacks. When an object stops moving, it is no longer detected, and once it starts moving again, its reappearance can produce a negative ghost that disrupts the analysis. The technique is also sensitive to the direction of motion. It highlights only those parts of an object that move against the surrounding background. If an edge travels in the same direction as the motion field, it fails to appear. As a result, only a portion of the object's outline is captured, typically the leading and trailing edges rather than the full silhouette.

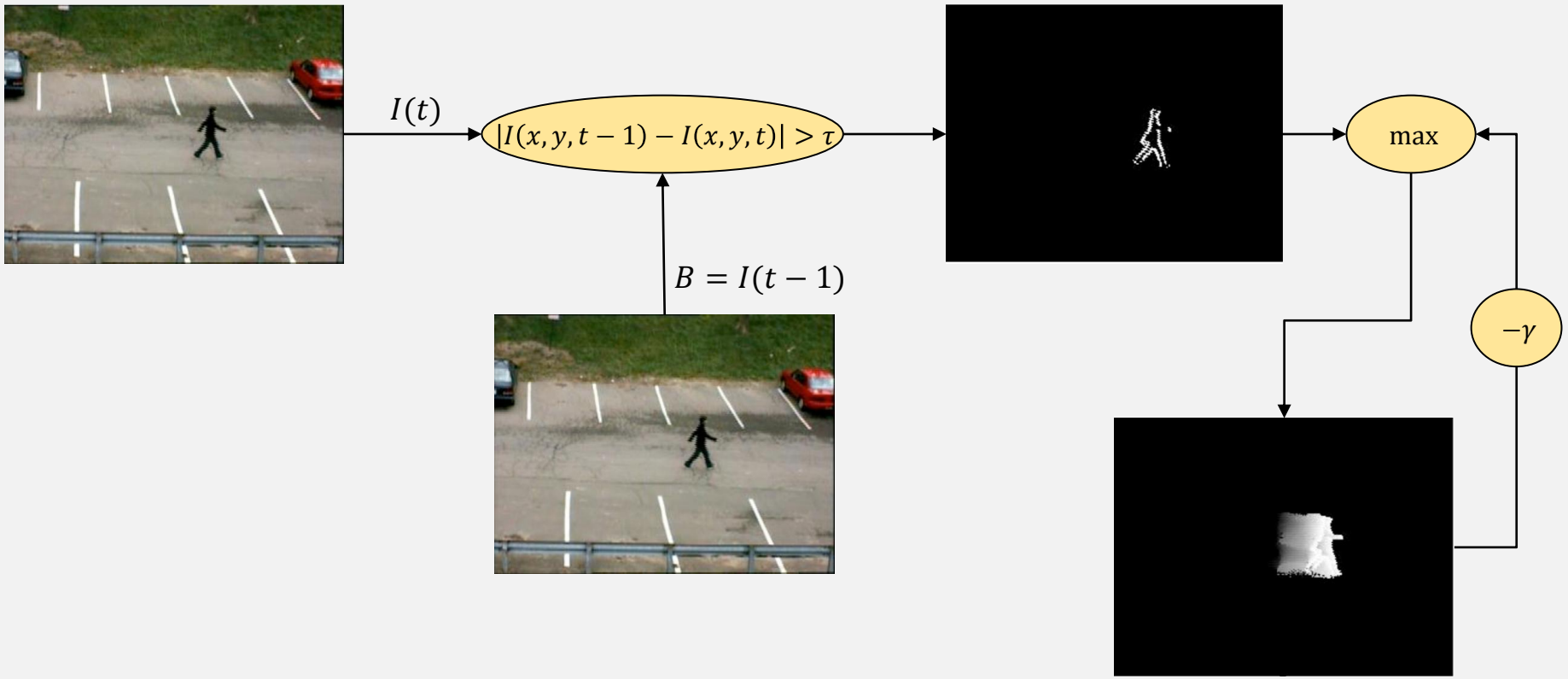
- **Three Frame Differencing:** In the previous method we used frame differencing between consecutive frames. Increasing the gap between frames gives clearer object outlines but creates two copies, showing where the object was and where it is now. To remove the extra copy, we compare three frames: the current frame, one from the past (for example 15 frames earlier) and one from the future (for example 15 frames later). We form two difference images and the area where they overlap indicates the object's current position. This approach introduces a delay in locating the object's current position because it uses a future frame.



The right frame-rate and time between images depend on how big and fast the objects are. In the example above, the current position is where the two difference images (one from the past, one from the future) intersect.

Images by Robert Collins

– **Motion History Images:** We calculate the changes between consecutive images to create a motion history. These are then blended with a gradual decrease over time. The motion history images show where the object has been.

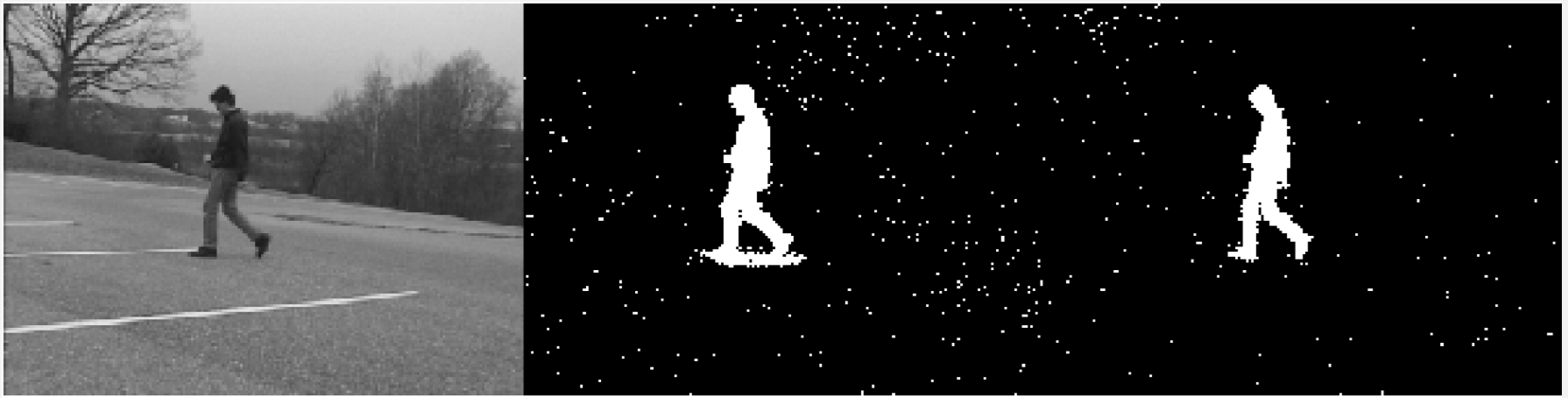


We create the current motion histogram by subtracting a selected value γ from the previous one (setting negative values to zero) and then combining it with the current motion image using the max function. The decay parameter adds gray values to the motion history, indicating how long ago the motion was detected (darker means earlier). A larger decay parameter means a shorter history of object motions. The motion history images summarize how much motion occurred over a specific 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 can detect moving objects and the shadows they create. However, shadows can lead to inaccurate location of the objects, the appearance of extra movement, and should be avoided. To tell the difference between a shadow and the actual object, we can compare the chromaticity (without brightness) of the pixels. A shadow will only change the brightness of the background color, but the chromaticity stays about the same. We can thus improve the differencing method used in the previous methods.
  - Instead of highlighting the differences between the RGB-values of pixels in subsequent images, we start by mapping the pixel values to a chromaticity sub-space (like a*b* or HS) without considering the brightness of pixels. Then, we use thresholding to remove shadows while preserving objects that are noticeably different in chromaticity from the background.

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



- **Optical Flow:** The earlier methods work only for cameras that do not move and cannot zoom or tilt. Detecting motion in general video requires different techniques. The most widely used method is the Lucas-Kanade algorithm.
  - The main idea is that the brightness of a pixel should stay the same over short periods of time, without sudden changes. Hence, we're looking at how a pixel's brightness changes as it moves and approximately set the differences over short time periods to 0. So, if we use $I(x(t), y(t), t)$ to show the brightness of a pixel and describe its path $[x(t), y(t)]$ as a function over time, brightness constancy means:

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

  - We will follow the movement of the pixel from one frame to the next, with a small time difference of Δt (e.g., 0.04 seconds with 25 frames per second). The pixel has shifted to a new position $x(t + \Delta t) = x(t) + u$ and $y(t + \Delta t) = y(t) + v$. We will use a Taylor expansion to linearize the equation and to solve for u and v:

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

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

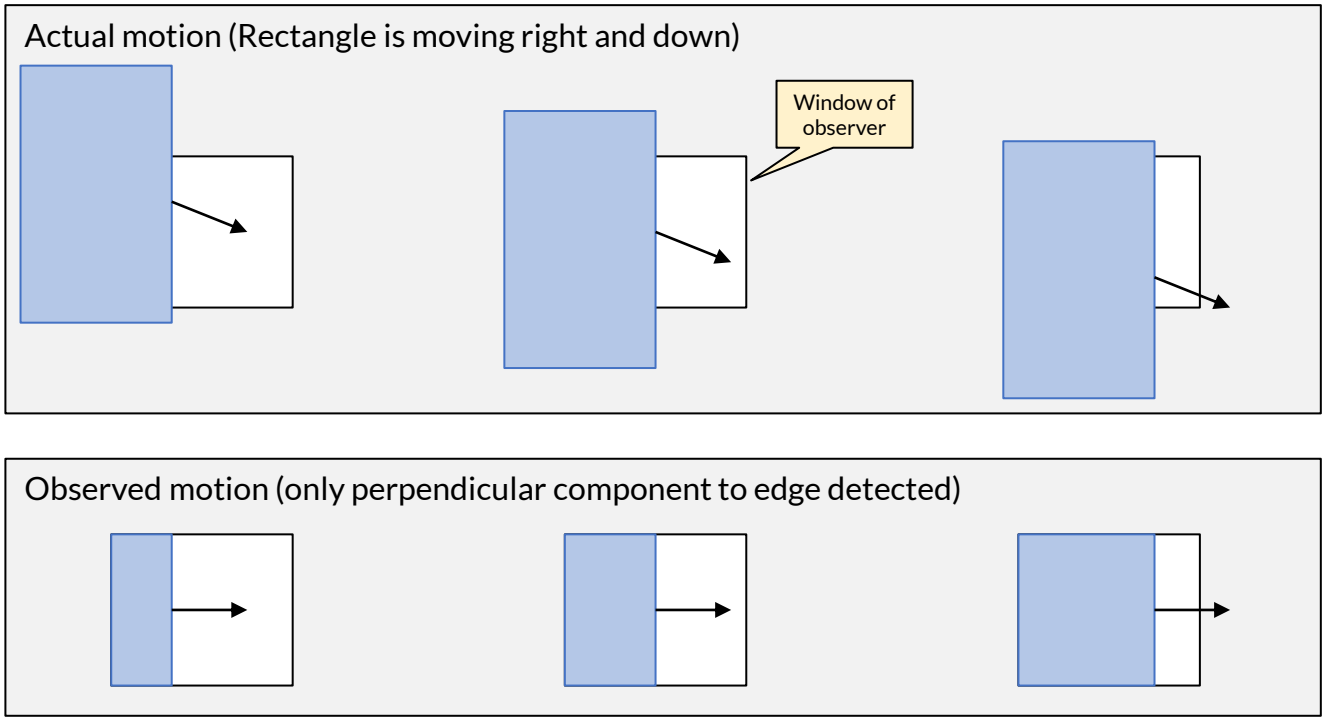
    $$0 \approx \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}\Delta t$$

  - 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 $t + \Delta t$ for the fixed position (x, y).

– We obtain the final equation for our motion estimate as follows:

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

We can observe the partial derivatives I_x, I_y and I_t from the changes in the frames, and we can solve with that the above linear equation for u and v to obtain the motion vector (u, v) for the pixel at (x, y) and time t. If we consider only one pixel, we have just one equation with two unknowns resulting in many possible solutions. If (u, v) is a solution, then $(u + u', v + v')$ is a solution with (u', v') 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**:



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

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

- The Lucas-Kanade method solved the aperture problem by looking at a 5x5 window around the current pixel. It assumed that the motion vectors for pixels in such a small window are about the same. This resulted in 25 equations for the two unknowns.

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

$$\begin{bmatrix} I_x(x-2,y-2) & I_y(x-2,y-2) \\ \vdots & \vdots \\ I_x(x,y) & I_y(x,y) \\ \vdots & \vdots \\ I_x(x+2,y+2) & I_y(x+2,y+2) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_t(x-2,y-2) \\ \vdots \\ -I_t(x,y) \\ \vdots \\ -I_t(x+2,y+2) \end{bmatrix}$$

$$\mathbf{A} \qquad \boldsymbol{d} = \qquad \boldsymbol{b}$$

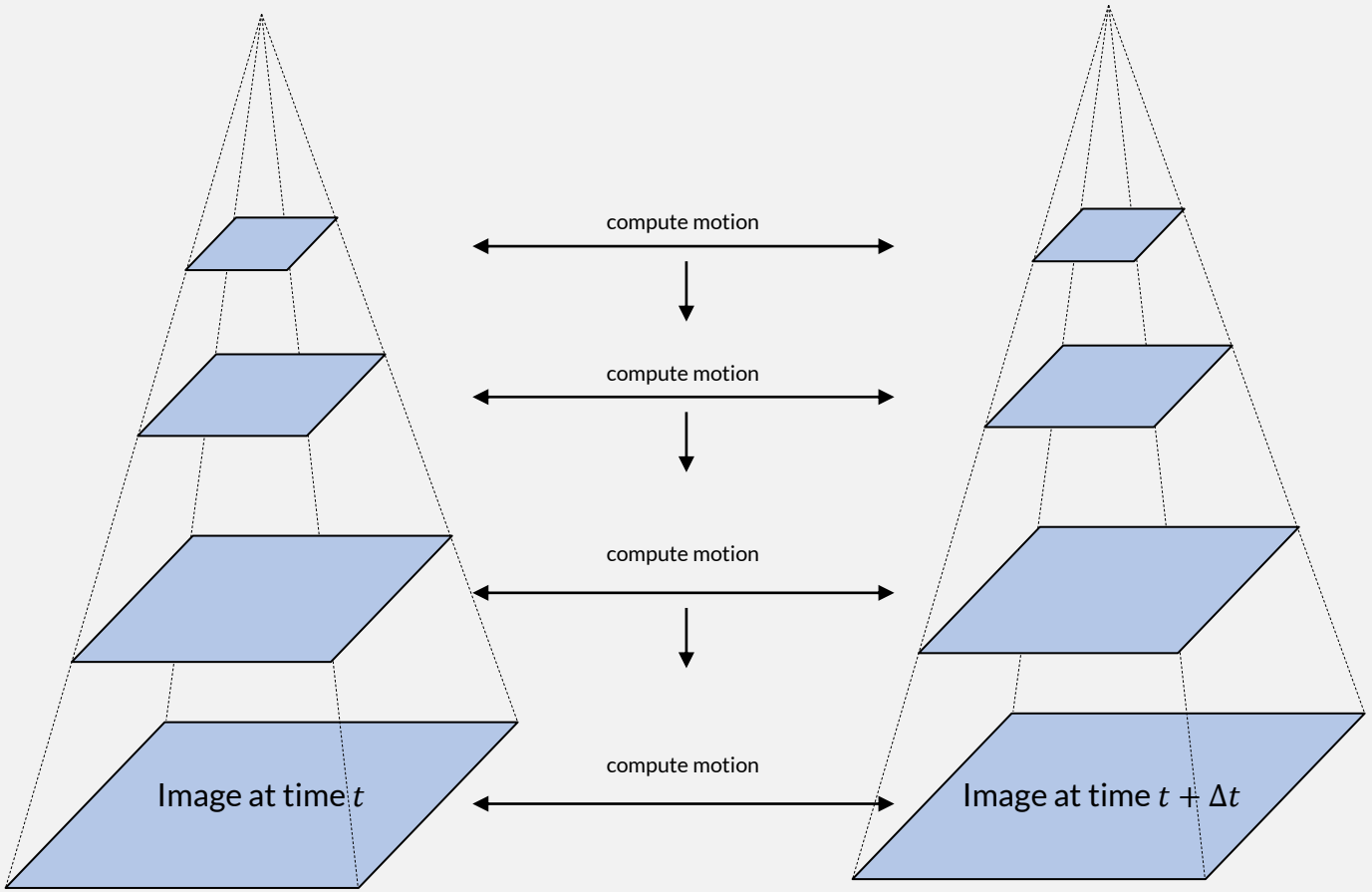
- Because there are more equations than unknowns, there is no exact answer. Instead, we minimize $\|\boldsymbol{A}\boldsymbol{d} - \boldsymbol{b}\|^2$ by solving its gradient for zero, which leads to:

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

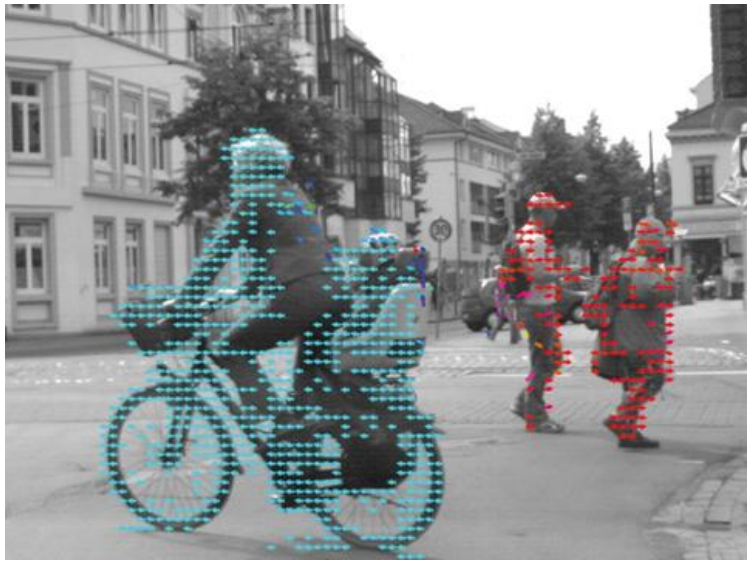
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

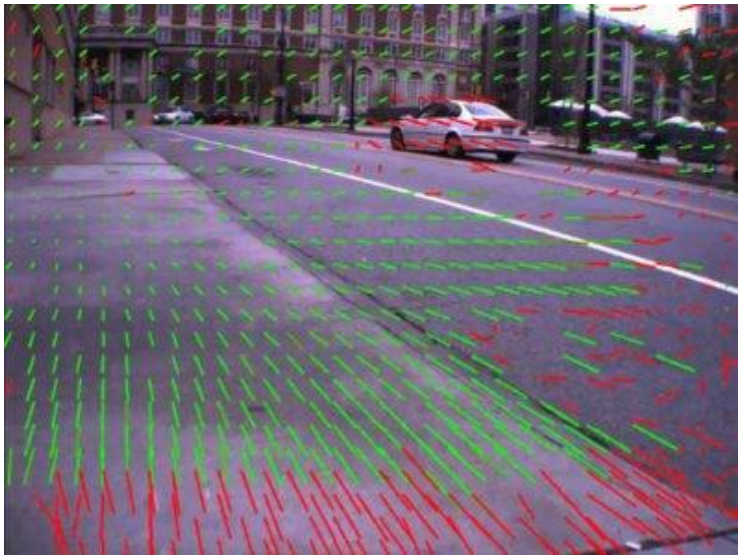
The sums are calculated for all the pixels in the 5x5 window around the current point. To solve the equation, the $\mathbf{A}^\top\mathbf{A}$ should be invertible, should not have very small eigenvalues, and the ratio between the two eigenvalues should not be too big. Therefore, this method works best for corner points (also known as key points or Harris points) but not for edge points or points in flat areas.

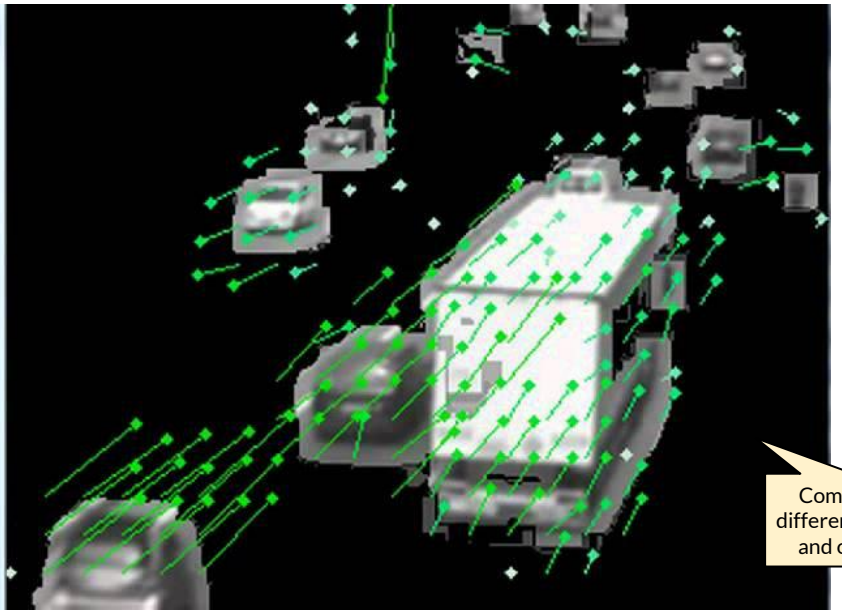
– The basic method is effective for small movements. For larger movements, we can use a Gaussian pyramid of the image and estimate flows at each level.



– Optical Flow: Examples (various sources)







13.4 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/